

**IMPLEMENTASI SISTEM *MONITORING* DAYA LISTRIK  
BERBASIS *WEB* DAN PROTOKOL KOMUNIKASI WEBSOCKET**

**SKRIPSI**

**KEMINATAN TEKNIK KOMPUTER**

Untuk memenuhi sebagian persyaratan  
Memperoleh gelar Sarjana Komputer

Disusun oleh:  
Zakky Ramadhan  
NIM: 135150301111016



PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
MALANG  
2018

## PENGESAHAN

### IMPLEMENTASI SISTEM *MONITORING* DAYA LISTRIK BERBASIS *WEB* DAN PROTOKOL KOMUNIKASI *WEBSOCKET*

#### SKRIPSI

#### KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun Oleh :

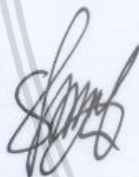
Zakky Ramadhan

NIM: 135150301111016

Skrripsi ini telah diuji dan dinyatakan lulus pada  
2 Agustus 2018

Telah diperiksa dan disetujui oleh:

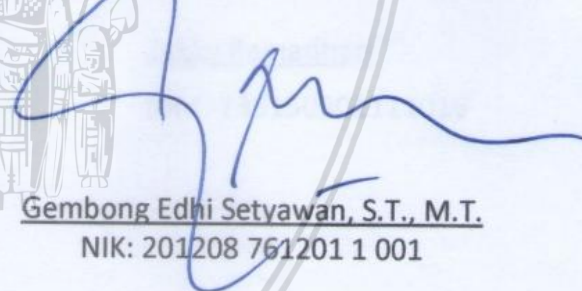
Dosen Pembimbing I



Sabriansyah Rizqika Akbar, S.T., M.Eng.

NIP: 19820809 201212 1 004

Dosen Pembimbing II



Gembong Edhi Setyawan, S.T., M.T.

NIK: 201208 761201 1 001

Mengetahui

Ketua Jurusan Teknik Informatika



Dr. Astoto Kurniawan, S.T., M.T., Ph.D

NIP: 19710518 200312 1 001

## PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018



Zakky Ramadhan

NIM: 135150301111016



## KATA PENGANTAR

Puji dan syukur penulis ucapkan kepada Tuhan Yang Maha Kuasa, karena atas limpahan rahmat dan karunia-Nya maka penulis dapat menyelesaikan laporan tugas akhir skripsi ini. Adapun maksud penyelesaian skripsi ini adalah untuk memenuhi persyaratan dalam menempuh ujian Sarjana Fakultas Ilmu Komputer, Universitas Brawijaya Malang. Judul skripsi yang dikerjakan oleh penulis adalah “Implementasi Sistem Monitoring Daya Listrik Berbasis *Web* dan Protokol Komunikasi *Websocket*”.

Penulis menyadari bahwa dalam proses pengerjaan skripsi dan penyusunan laporan skripsi ini tidak akan terwujud tanpa adanya bantuan dan dorongan dari berbagai pihak. Oleh karena itu, maka pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

1. Kedua orang tua serta keluarga penulis yang sudah memberikan doa dan dukungan kepada penulis.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T., Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng. selaku Ketua Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Brawijaya Malang sekaligus Dosen Pembimbing I skripsi ini.
5. Bapak Gembong Edhi Setyawan, S.T., M.T. selaku Dosen Pembimbing Akademik sekaligus Dosen Pembimbing II skripsi ini.
6. Seluruh teman-teman tercinta yang telah membantu proses penyelesaian skripsi ini yang mana tidak dapat saya sebutkan satu persatu.

Akhir kata, penulis berharap semoga laporan skripsi ini dapat bermanfaat bagi seluruh pihak dan dapat menjadi acuan untuk pengembangan penelitian berikutnya.

Malang, 2 Agustus 2018

Zakky Ramadhan

zakkyramadhan@yahoo.co.id



## ABSTRAK

Penggunaan dan biaya energi listrik yang setiap tahunnya semakin meningkat mendorong masyarakat untuk mengontrol penggunaan energi listrik. Salah satu langkah awal yang dilakukan dalam rangka mengontrol penggunaan energi listrik adalah dengan melakukan *monitoring* terhadap perangkat elektronik yang digunakan. Pada penelitian ini, dikembangkan sebuah sistem *monitoring* daya listrik yang dipasang pada terminal listrik di sebuah ruangan dengan menggunakan sensor arus *current transformer* (CT) sensor YHDC SCT-013-000 dan mikrokontroler NodeMCU. Nantinya, perangkat listrik yang terhubung pada terminal listrik tersebut akan dipantau penggunaan daya listriknya. Data *monitoring* daya listrik dari perangkat *monitoring* akan dikirimkan dengan menggunakan protokol komunikasi Websocket yang selanjutnya akan disimpan pada *database server*. Sedangkan untuk data *monitoring* daya listrik ditampilkan pada antarmuka *web*. Dengan begitu, diharapkan pengguna dapat melakukan *monitoring* daya listrik pada ruangan dengan lebih mudah dan akurat. Hal ini berdasarkan hasil pengujian yang telah dilakukan, didapat akurasi pengukuran arus listrik hingga 97,14%, akurasi tegangan listrik hingga 99%, dan akurasi perhitungan daya listrik mencapai 98%. Untuk performa penerimaan data *monitoring* daya listrik yang didapat membutuhkan waktu rata-rata sebesar 160,8 milidetik. Sedangkan untuk *web monitoring* daya listrik yang dibangun mampu menampilkan penggunaan daya listrik hingga estimasi biaya penggunaan listrik pada setiap perangkat *monitoring*.

Kata kunci: CT sensor, Listrik, *Monitoring*, NodeMCU, Websocket, *Web*

## ABSTRACT

*Electricity usage and its price are always increased every year which encourages people to control their electricity usage. We can control electricity usage by doing electricity usage monitoring. In this study, we developed an electric power monitoring system which installed on the power strip using current transformer sensor YHDC SCT-013-000 and NodeMCU microcontroller. So, electric power usage of electronic devices that plugged into the power strip will be monitored. For data transmission, we used Websocket protocol for communicating all monitoring devices to the server. All monitoring data is saved to database server and electricity monitoring output will be showed using web interface. We hope users can monitor their electricity usage easier and more accurate. Based on the test, we get up to 97,14% accuracy result for electric current, up to 99% for electric voltage, and up to 98% accuracy result for electric power calculation. We also get a good performance with an average of 160,8 milliseconds for data monitoring acquisition. For the monitoring web, we built a web which has features such as showing electric power usage and estimated power cost on each monitoring devices.*

*Keywords: CT Sensor, Electricity, Monitoring, NodeMCU, Websocket, Web*



## DAFTAR ISI

PENGESAHAN .....	ii
PERNYATAAN ORISINALITAS .....	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	v
ABSTRACT .....	vi
DAFTAR ISI .....	vii
DAFTAR TABEL.....	x
DAFTAR GAMBAR.....	xi
BAB 1 PENDAHULUAN.....	1
1.1 Latar belakang.....	1
1.2 Rumusan masalah.....	3
1.3 Tujuan .....	4
1.4 Manfaat.....	4
1.5 Batasan masalah .....	4
1.6 Sistematika pembahasan.....	5
BAB 2 LANDASAN KEPUSTAKAAN .....	7
2.1 Kajian Pustaka .....	7
2.2 Dasar Teori.....	8
2.2.1 Mikrokontroler .....	8
2.2.2 <i>Analog-to-Digital Converter</i> .....	8
2.2.3 Rangkaian Pembagi Tegangan .....	8
2.2.4 Sensor Arus <i>Current Transformer</i> Sensor YHDC SCT-013-000 ....	9
2.2.5 Tarif Dasar Listrik.....	11
2.2.6 Protokol Websocket.....	12
2.2.7 Node.js .....	13
2.2.8 JSON .....	14
2.2.9 <i>OpenEnergyMonitor</i> .....	14
2.2.10 <i>Framework</i> PHP CodeIgniter .....	14
BAB 3 METODOLOGI .....	16
3.1 Metodologi Penelitian .....	16



3.2 Studi Literatur .....	16
3.3 Analisis Kebutuhan Sistem.....	17
3.3.1 Kebutuhan Perangkat Keras.....	17
3.3.2 Kebutuhan Perangkat Lunak .....	17
3.4 Perancangan Sistem.....	17
3.5 Implementasi Sistem .....	18
3.6 Pengujian Sistem.....	19
3.7 Kesimpulan dan Saran .....	19
BAB 4 ANALISIS KEBUTUHAN .....	20
4.1 Kebutuhan Pengguna.....	20
4.2 Kebutuhan Sistem .....	21
4.2.1 Kebutuhan Perangkat Keras.....	21
4.2.2 Kebutuhan Perangkat Lunak .....	24
4.3 Kebutuhan Fungsional .....	25
4.4 Kebutuhan Non Fungsional.....	26
4.4.1 Karakteristik Pengguna .....	26
4.4.2 Lingkungan Pengguna .....	26
4.4.3 Batasan Desain Sistem .....	26
BAB 5 PERANCANGAN DAN IMPLEMENTASI .....	27
5.1 Perancangan Sistem.....	27
5.1.1 Perancangan Perangkat Keras .....	28
5.1.2 Perancangan Perangkat Lunak.....	30
5.1.3 Perancangan Keseluruhan Sistem.....	45
5.2 Implementasi .....	46
5.2.1 Implementasi Perangkat Keras .....	47
5.2.2 Implementasi Perangkat Lunak.....	49
BAB 6 PENGUJIAN DAN ANALISIS.....	73
6.1 Pengujian Perangkat Keras .....	73
6.1.1 Pengujian Akurasi <i>Monitoring</i> Perangkat Keras .....	73
6.2 Pengujian Perangkat Lunak.....	77
6.2.1 Pengujian Performa Penerimaan Data <i>Monitoring</i> Listrik.....	77
6.2.2 Pengujian Fungsionalitas <i>Web Monitoring</i> Daya Listrik .....	79

BAB 7 PENUTUP .....	82
7.1 Kesimpulan.....	82
7.2 Saran .....	83
Daftar Pustaka .....	84



## DAFTAR TABEL

Tabel 2.1 Spesifikasi CT Sensor YHDC SCT-013-000 .....	10
Tabel 4.1 Spesifikasi Mikrokontroler NodeMCU.....	22
Tabel 4.2 Spesifikasi Taff <i>Energy Power Meter</i> .....	24
Tabel 5.1 Tabel <i>monitoring</i> .....	36
Tabel 5.2 Tabel Ruangan .....	38
Tabel 5.3 Tabel list_device .....	38
Tabel 5.4 Tabel TDL .....	39
Tabel 5.5 Potongan kode sumber kategori inisialisasi awal .....	49
Tabel 5.6 Potongan kode sumber kategori Websocket.....	51
Tabel 5.7 Potongan kode sumber kategori baca arus .....	52
Tabel 5.8 Potongan kode sumber inisialisasi <i>server dan penanggalan</i> .....	55
Tabel 5.9 Potongan kode sumber Websocket pada Websocket <i>server</i> .....	58
Tabel 6.1 Ruang Pengujian .....	74
Tabel 6.2 Pengujian Akurasi Perangkat Keras <i>Monitoring</i> Listrik .....	75
Tabel 6.3 Performa Penerimaan Data <i>Monitoring</i> Listrik .....	78
Tabel 6.4 Hasil Pengujian Fungsionalitas <i>Web Monitoring listrik</i> .....	80



## DAFTAR GAMBAR

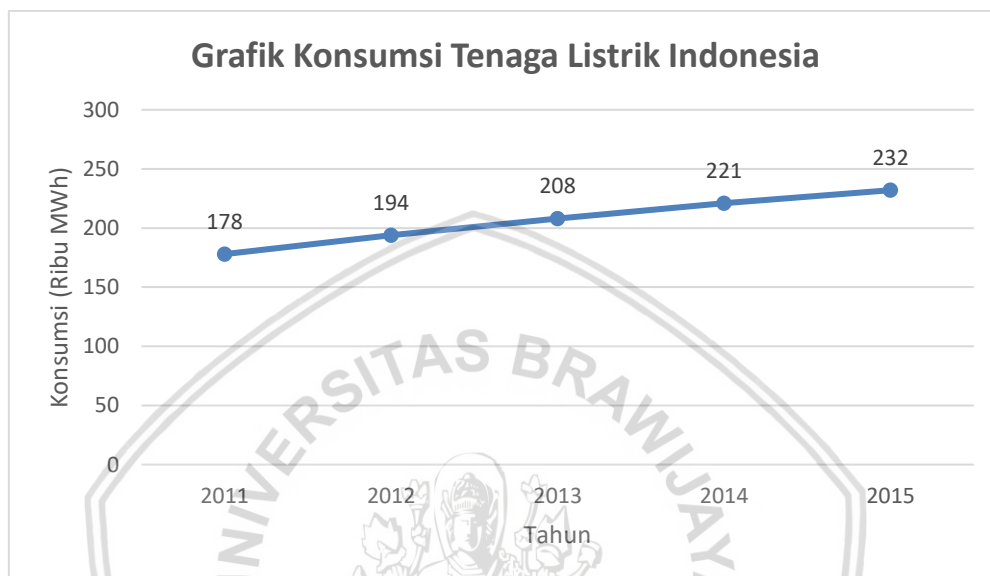
Gambar 1.1 Grafik Konsumsi Tenaga Listrik Indonesia.....	1
Gambar 1.2 Grafik Kapasitas Terpasang Pembangkit Tenaga Listrik Nasional.....	2
Gambar 1.3 Contoh struk pembayaran listrik.....	2
Gambar 2.1 Rangkaian Pembagi Tegangan Sederhana .....	9
Gambar 2.2 Sensor Arus <i>Current Transformer Sensor</i> YHDC SCT-013-000 .....	10
Gambar 2.3 Daftar TDL Bulan Juli-September 2017 .....	12
Gambar 2.4 Perbedaan AJAX dengan Websocket .....	13
Gambar 2.5 Arsitektur dari Node.js .....	13
Gambar 2.6 Format Penulisan Objek JSON.....	14
Gambar 2.7 Arsitektur Sederhana dari Konsep MVC.....	15
Gambar 3.1 Diagram Alir Metode Penelitian.....	16
Gambar 3.2 Diagram Blok Perancangan <i>Input, Proses, dan Output</i> .....	18
Gambar 4.1 Diagram Analisa Kebutuhan .....	20
Gambar 4.2 <i>Laptop</i> Dell Inspiron 14 7447 Pandora.....	21
Gambar 4.3 <i>Wireless router</i> Netgear DGN1000.....	22
Gambar 4.4 Mikrokontroler NodeMCU .....	22
Gambar 4.5 <i>Taff Energy Power Meter</i> .....	24
Gambar 5.1 Struktur Perancangan Sistem.....	27
Gambar 5.2 Skematik Perangkat Keras.....	28
Gambar 5.3 Pengkondisian Sinyal <i>Analog</i> Listrik oleh Sensor Arus.....	30
Gambar 5.4 <i>Flowchart</i> proses inisialisasi awal .....	31
Gambar 5.5 <i>Flowchart</i> Websocket.....	32
Gambar 5.6 <i>Flowchart</i> baca arus .....	33
Gambar 5.7 <i>Flowchart</i> Websocket server.....	35
Gambar 5.8 Desain Relasi <i>Database</i> .....	36
Gambar 5.9 <i>Flowchart</i> Data Ruangan .....	41
Gambar 5.10 <i>Flowchart</i> Perangkat Baru dan Perangkat Terdaftar .....	42
Gambar 5.11 <i>Flowchart Monitoring</i> .....	43
Gambar 5.12 <i>Flowchart Detail Monitoring</i> .....	44
Gambar 5.13 <i>Flowchart</i> Tarif Dasar Listrik.....	45

Gambar 5.14 Alur Kerja Keseluruhan Sistem .....	46
Gambar 5.15 Bentuk fisik perangkat keras .....	47
Gambar 5.16 Perangkat keras terpasang pada terminal listrik .....	48
Gambar 5.17 <i>Output monitoring</i> pada <i>serial monitor</i> Arduino IDE.....	48
Gambar 5.18 Proses insialisasi awal node.js Websocket <i>server</i> .....	54
Gambar 5.19 Pemasangan modul Websocket.....	54
Gambar 5.20 Pemasangan modul nodemon .....	55
Gambar 5.21 Websocket <i>server</i> node.js yang sedang berjalan .....	62
Gambar 5.22 <i>Database Server</i> Sistem <i>Monitoring</i> Listrik.....	62
Gambar 5.23 Tabel <i>list_device</i> di <i>database server</i> .....	63
Gambar 5.24 Tabel <i>monitoring</i> di <i>database server</i> .....	63
Gambar 5.25 Tabel ruangan di <i>database server</i> .....	64
Gambar 5.26 Tabel TDL pada <i>database server</i> .....	64
Gambar 5.27 Tabel <i>temp_mac</i> pada <i>database server</i> .....	65
Gambar 5.28 Halaman Ruangan .....	66
Gambar 5.29 Daftar Data Ruangan.....	66
Gambar 5.30 Daftar Perangkat Baru.....	67
Gambar 5.31 Daftar Perangkat Terdaftar .....	68
Gambar 5.32 Halaman <i>Monitoring</i> .....	68
Gambar 5.33 Grafik Penggunaan Listrik .....	69
Gambar 5.34 Tabel Penggunaan Listrik .....	69
Gambar 5.35 Grafik Total Penggunaan Listrik .....	70
Gambar 5.36 Tabel Total Penggunaan Listrik .....	70
Gambar 5.37 Halaman Detail <i>Monitoring</i> .....	71
Gambar 5.38 Halaman Tarif Dasar Listrik .....	71
Gambar 5.39 <i>Web Monitoring</i> Daya Listrik Diakses <i>Smartphone</i> .....	72
Gambar 6.1 <i>Flowchart</i> Pengujian dan Analisa .....	73
Gambar 6.2 Grafik Perbandingan Pengukuran Arus Listrik .....	75
Gambar 6.3 Grafik Perbandingan Pengukuran Tegangan Listrik.....	76
Gambar 6.4 Grafik Perbandingan Pengukuran Daya Listrik.....	76
Gambar 6.5 Grafik Akurasi <i>Monitoring</i> Perangkat Keras.....	77
Gambar 6.6 Grafik Performa Penerimaan Data <i>Monitoring</i> Listrik .....	79

## BAB 1 PENDAHULUAN

### 1.1 Latar belakang

Salah satu tanda berkembangnya kehidupan manusia dapat dilihat dari penggunaan energi khususnya energi listrik yang dari tahun ke tahun jumlahnya semakin meningkat seperti yang ditampilkan pada Gambar 1.1.



**Gambar 1.1** Grafik Konsumsi Tenaga Listrik Indonesia

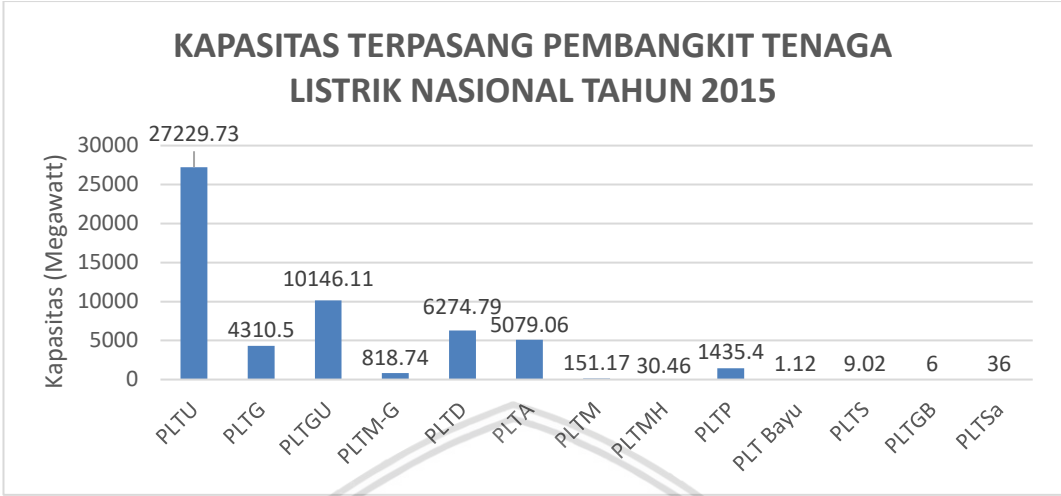
**Sumber:** (Direktorat Jenderal Ketenagalistrikan, 2016)

Listrik memegang peranan penting dalam kehidupan manusia karena sebagian besar peralatan yang digunakan oleh manusia untuk menyelesaikan sebuah pekerjaan menggunakan energi listrik. Namun, meningkatnya penggunaan energi listrik yang terjadi setiap harinya juga menyebabkan meningkatnya penggunaan sumber daya yang digunakan untuk menghasilkan energi listrik tersebut. Sebagian besar energi listrik di Indonesia masih dihasilkan dari pembangkit listrik yang menggunakan batu bara, gas, dan bahan bakar fosil (diesel) dimana bahan-bahan tersebut merupakan sumber daya alam yang tidak terbarukan, seperti yang diperlihatkan pada Gambar 1.2. Hal tersebut mendorong pentingnya melakukan pemantauan dan penghematan penggunaan listrik serta penggunaan energi terbarukan untuk menghasilkan energi listrik seperti misalnya tenaga surya, angin, air, dan lain-lain.

Pemerintah pada beberapa tahun belakangan pernah mencanangkan "Gerakan Hemat Listrik 17-22" dimana pemerintah mengajak masyarakat untuk menghemat penggunaan listrik terutama pada pukul 17.00 hingga pukul 22.00 (Christina, 2012). Komunitas Earth Hour mengkampanyekan "Gerakan Matikan Lampu Selama Satu Jam" tiap tahun di seluruh dunia pada hari bumi merupakan salah satu upaya dalam mengurangi penggunaan energi listrik (WWF Indonesia, 2016). Oleh karena itu, menghemat penggunaan listrik merupakan hal yang

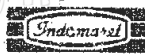


penting dilakukan dan langkah pertama yang dilakukan adalah dengan melakukan *monitoring* pada penggunaannya.



**Gambar 1.2** Grafik Kapasitas Terpasang Pembangkit Tenaga Listrik Nasional  
**Sumber:** (Direktorat Jenderal Ketenagalistrikan, 2016)

Pengguna listrik rumahan saat ini hanya dapat melakukan *monitoring* penggunaan daya listrik melalui KWh meter dimana data penggunaan listrik dapat dilihat pada struk pembayaran listrik bulanan berupa jumlah biaya listrik yang harus dibayar, namun tidak ada berapa jumlah KWh listrik yang telah digunakan selama satu bulan, seperti yang diperlihatkan pada Gambar 1.3. Jumlah biaya listrik yang harus dibayar tersebut merupakan jumlah pemakaian KWh seluruh perangkat listrik yang ada di rumah dikalikan dengan biaya TDL (Tarif Dasar Listrik) yang sudah ditentukan oleh PLN sesuai dengan golongan TDL tersebut (Listrik.org, 2017).



KARANG WIDORO, MALANG/ 08865523480  
JL. RAYA CANDI V KARANG WIDORO, MALANG, 65146

BANK MANDIRI

STRUK PEMBAYARAN TAGIHAN LISTRIK

ICPEL : 513110554319  
NAMA : AGUS PRASMONO  
TARIF/DAYA : R1M/900VA  
BL/TH : MEI17  
STAND METER : 01836400-01849600  
RP TAG PLN : Rp 153,489  
JPA REF : 0IND210ZEE42E00A  
645FAD9C85E864CC

PLN menyatakan struk ini sbg bukti pembayaran yg sah.

ADMIN BANK : Rp 2,000  
TOTAL BAYAR : Rp 155,489

**Gambar 1.3** Contoh struk pembayaran listrik

Penelitian sebelumnya yang dilakukan oleh Anggraeni (2016) telah mengembangkan sebuah sistem *monitoring* listrik menggunakan mikrokontroler AVR ATmega 8535 dan sensor arus listrik ACS712 30 A. *Output monitoring* listrik hanya berupa penggunaan daya listrik dari perangkat elektronik pada saat itu yang ditampilkan pada layar LCD.

Pada penelitian berikutnya yang dilakukan oleh Kurniawan (2017), telah dikembangkan sebuah sistem *monitoring* listrik dengan menggunakan mikrokontroler NodeMCU dan sensor arus *current transformer sensor*. Sistem *monitoring* tersebut berbasis aplikasi dengan komunikasi menggunakan protokol MQTT yang disimpan pada *broker* Thingspeak. Untuk *output monitoring* listrik hanya sebatas penggunaan daya listrik dari perangkat elektronik pada waktu tertentu yang diakses melalui aplikasi Thingspeak di *smartphone*.

Umumnya pada setiap ruangan dalam suatu rumah hanya tersedia sebuah stop kontak. Maka dari itu, masyarakat biasanya menggunakan terminal listrik untuk menghubungkan satu atau beberapa perangkat listrik ke sebuah stop kontak. Terminal listrik tersebut dapat dipasang sebuah perangkat *monitoring* arus listrik dengan memanfaatkan mikrokontroler NodeMCU dan sensor arus listrik *current transformer (CT) sensor* YHDC SCT-013-000 untuk melakukan *monitoring* penggunaan listrik di ruangan. Mikrokontroler NodeMCU memiliki keunggulan dimana NodeMCU memiliki bentuk fisik yang ringkas, sudah terintegrasi dengan Wi-Fi, dan terdapat sebuah *input analog* untuk menghubungkan sensor arus listrik. Sensor YHDC SCT-013-000 juga memiliki keunggulan pada pembacaan nilai arus listrik hingga 100 A serta pemasangannya yang cukup mudah hanya dengan menjepit sensor tersebut pada salah satu kabel terminal listrik. Untuk pengiriman data, peneliti menggunakan protokol komunikasi Websocket karena memiliki karakteristik *low latency* pada penggunaan *traffic* jaringan serta bekerja dalam mode *full duplex* dibanding dengan menggunakan HTTP-AJAX pada umumnya yang masih bekerja dalam mode *half duplex* (Sim., 2014). Jadi, antara perangkat *monitoring* dengan *server* dapat berkomunikasi dua arah secara simultan. Sedangkan untuk *output monitoring* daya listrik akan ditampilkan dalam antarmuka *web* karena bersifat *cross-platform* dapat diakses dari *browser* pada komputer maupun *smartphone*.

Diharapkan dengan adanya penelitian ini, dapat memudahkan masyarakat untuk melakukan *monitoring* terhadap penggunaan energi listrik di setiap ruangan yang ada di rumah.

## 1.2 Rumusan masalah

1. Bagaimana akurasi dari perangkat keras *monitoring* daya listrik yang dibuat dalam melakukan *monitoring* penggunaan listrik di tiap ruangan yang ada pada sebuah rumah?
2. Bagaimana performa penerimaan data *monitoring* listrik pada sistem *monitoring* daya listrik yang dibangun?

3. Bagaimana kinerja fungsionalitas dari *web monitoring* daya listrik yang dibangun?

### 1.3 Tujuan

Adapun tujuan dari dilakukannya penelitian ini diantaranya:

1. Merancang sebuah sistem *monitoring* daya listrik yang memiliki keakuratan pengukuran yang tinggi baik dalam mengukur arus, tegangan, dan daya listrik.
2. Membangun sebuah sistem *monitoring* daya listrik yang memiliki performa akuisisi data *monitoring* yang baik pada setiap data *monitoring* yang didapat.
3. Membangun antarmuka *web monitoring* daya listrik yang bersifat *user friendly* dan memiliki fitur-fitur yang dapat memudahkan pengguna dalam melakukan *monitoring* penggunaan daya listrik.

### 1.4 Manfaat

Manfaat yang diperoleh dari dilakukannya penelitian ini diantaranya:

1. Membantu pengguna dalam melakukan *monitoring* penggunaan listrik pada terminal listrik yang ada di ruangan.
2. Membantu pengguna dalam rangka melakukan pemantauan pada penggunaan energi listrik di tiap ruangan karena setiap ruangan dapat dipantau penggunaan listriknya.
3. Memudahkan dalam melakukan *monitoring* penggunaan daya listrik karena antarmuka sistem yang berbasis *web* dapat diakses melalui *browser* di komputer atau *smartphone*.

### 1.5 Batasan masalah

Untuk memudahkan penelitian, maka penulis membatasi ruang lingkup penelitian ini dengan batasan sebagai berikut:

1. Mikrokontroler yang digunakan pada penelitian ini adalah mikrokontroler NodeMCU.
2. Sensor arus listrik yang digunakan sensor AC *current sensor* YHDC SCT-013-000 100A yang digunakan untuk melakukan *monitoring* penggunaan daya listrik.
3. Sistem *monitoring* daya listrik diterapkan pada jaringan lokal.
4. Protokol komunikasi yang digunakan adalah protokol komunikasi Websocket.
5. Antarmuka sistem menggunakan *web* berbasis *framework* PHP CodeIgniter.
6. Fokus penelitian ini adalah sensor berhasil melakukan akuisisi data arus listrik, tersampainya data *monitoring* dari perangkat *monitoring* ke Websocket *server* yang selanjutnya disimpan pada *database server*, dan *web monitoring* berhasil menampilkan data *monitoring* penggunaan listrik pada setiap perangkat *monitoring* yang terpasang.



## 1.6 Sistematika pembahasan

Sistematika pengelompokan materi yang dilakukan dalam penyusunan laporan akhir ini dibagi dalam beberapa sub bab penelitian yang diuraikan sebagai berikut:

### **BAB 1 PENDAHULUAN**

Pada pendahuluan mencakup latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika rencana dari penelitian “Implementasi Sistem *Monitoring* Daya Listrik Berbasis Web dan Protokol Komunikasi Websocket”

### **BAB 2 LANDASAN KEPUSTAKAAN**

Bab ini berisi kajian tentang penelitian sebelumnya yang terkait dengan sistem *monitoring* listrik dan memaparkan teori-teori serta konsep-konsep yang didapat dari sumber-sumber atau penelitian yang relevan untuk digunakan sebagai panduan dalam penelitian serta penyusunan laporan.

### **BAB 3 METODOLOGI**

Bab ini membahas metodologi yang digunakan pada penelitian ini, berisi pemaparan langkah kerja yang terdiri atas landasan kepastakaan, analisis kebutuhan, perancangan dan implementasi sistem, analisis dan pengujian, serta penarikan kesimpulan dan saran untuk penelitian berikutnya.

### **BAB 4 ANALISIS KEBUTUHAN**

Pada bab ini akan menjelaskan terkait kebutuhan apa saja yang diperlukan dalam menyelesaikan penelitian, yang dibagi dalam kebutuhan perangkat keras, kebutuhan perangkat lunak, kebutuhan fungsional, dan kebutuhan non-fungsional.

### **BAB 5 PERANCANGAN DAN IMPLEMENTASI**

Bab ini menguraikan perancangan dan implementasi dari sistem *monitoring* daya listrik yang dibangun. Sistem yang dibangun terdiri dari perancangan perangkat keras, perancangan Websocket server, perancangan database server, perancangan web *monitoring* daya listrik, serta implementasi dari perancangan-perancangan tersebut.

### **BAB 6 PENGUJIAN DAN ANALISIS**

Bab ini memuat hasil pengujian dan analisis dari sistem *monitoring* daya listrik yang dibangun. Bagian-bagian yang diuji dan dianalisa diantaranya akurasi pengukuran dari sensor arus listrik yang digunakan, performa penerimaan data *monitoring* daya listrik, serta fungsionalitas dari web *monitoring* daya listrik yang dibangun.

**BAB 7      PENUTUP**

Bab ini berisi kesimpulan atas penelitian yang telah dikerjakan serta memberi saran untuk mengembangkan lebih lanjut dari sistem yang dibangun



## BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepastakaan berisi uraian dan pembahasan tentang kajian pustaka dan dasar teori. Pada bagian kajian pustaka membahas penelitian yang sudah ada dan diusulkan. Sedangkan pada bagian dasar teori membahas teori yang dibutuhkan untuk menyelesaikan penelitian yang diusulkan oleh penulis.

### 2.1 Kajian Pustaka

Berikut ini beberapa penelitian terkait sistem *monitoring* listrik menggunakan metode dan alat yang berbeda-beda dalam membangun sebuah sistem *monitoring* listrik. Berdasarkan penelitian yang telah dilakukan tersebut, peneliti dapat menggunakan penelitian tersebut sebagai acuan dalam mengembangkan sistem yang akan dibangun.

Pada penelitian yang dilakukan oleh Kurniawan (2017), peneliti mengembangkan sebuah sistem *monitoring* listrik pada ruangan menggunakan mikrokontroler NodeMCU dan sensor arus CT sensor. Sistem *monitoring* listrik tersebut berbasis aplikasi dengan menggunakan protokol komunikasi MQTT dan *broker* Thingspeak. *Output monitoring* yang ditampilkan berupa penggunaan listrik dari sebuah perangkat pada waktu tertentu dalam satuan Watt yang dapat diakses dengan menggunakan aplikasi Thingspeak pada *smartphone*.

Penelitian berikutnya yang dilakukan oleh Anggraeni (2016), peneliti juga mengembangkan sebuah sistem *monitoring* listrik pada perangkat listrik menggunakan mikrokontroler AVR ATmega 8535 dan sensor arus listrik ACS712 30 A. Data penggunaan listrik ditampilkan pada layar LCD dalam satuan Watt.

Saha (2014) telah mengembangkan EnergyLens, yaitu sebuah sistem *monitoring* penggunaan energi listrik pada perangkat elektronik yang ada di rumah. EnergyLens memanfaatkan sensor yang terpasang pada KWh meter dan mikrofon *smartphone* pengguna untuk menentukan *output* berupa besarnya penggunaan energi listrik, perangkat elektronik yang aktif, dan siapa saja yang menggunakan perangkat elektronik pada waktu tertentu dengan menggunakan metode SVM (*Support Vector Machine*).

Lea (2016) juga mengembangkan sebuah *open-source project* yang dinamakan OpenEnergyMonitor, dimana salah satu hasil dari *project* tersebut adalah *perangkat monitoring* listrik berbasis Raspberry Pi yang diberi nama emonPi. emonPi menggunakan Raspberry Pi sebagai mikrokontroler dan sensor arus CT sensor 100 A. Untuk *output monitoring* berupa tampilan penggunaan energi listrik dan keadaan lingkungan sekitar seperti suhu dan kelembaban dengan menggunakan aplikasi *web open-source* buatan mereka yang diberi nama Emoncms.

## 2.2 Dasar Teori

Pada bagian ini akan membahas teori yang diperlukan dalam menyusun penelitian yang diusulkan.

### 2.2.1 Mikrokontroler

Mikrokontroler adalah sebuah sistem komputer dimana prosesor, memori, dan banyak komponen lainnya seperti *interface controller*, *timer*, *interrupt controller*, dan *general purpose I/O* (GPIO) terintegrasi dalam sebuah *chip*. Pada mikrokontroler sudah terdapat semua komponen yang membuatnya dapat beroperasi secara *stand-alone* dan didesain khusus untuk *monitoring* dan/atau *controlling* (Gridling & Weiss, 2007).

### 2.2.2 Analog-to-Digital Converter

*Analog-to-Digital Converter* atau biasa disingkat dengan ADC merupakan sebuah perangkat elektronik yang memiliki fungsi untuk mengubah sinyal *input analog* (sinyal kontinyu) yang biasa ditemui dalam kehidupan sehari-hari seperti suara, cahaya, listrik, dan lain-lain ke dalam bentuk sinyal *digital* yang umum digunakan dalam sistem komputer (Le, et al., 2005).

Terdapat dua karakter dalam ADC, yaitu kecepatan/rasio *sampling* (*sampling rate*) dan resolusi ADC. Kecepatan *sampling* menyatakan seberapa sering sinyal analog diubah ke dalam bentuk sinyal *digital* pada jumlah waktu tertentu. Kecepatan *sampling* ini biasa dinyatakan dalam bentuk *sample per second* (SPS). Sedangkan resolusi ADC menentukan akurasi ketelitian nilai hasil konversi atau pengubahan sinyal *analog* ke dalam bentuk diskrit sinyal *digital*. Besarnya nilai resolusi dapat dihitung dengan menggunakan persamaan  $2^n - 1$ , dimana  $n$  merupakan nilai bit output dari ADC tersebut. Sebagai contoh pada ADC dengan *output digital* 8-bit akan mengubah sinyal *input analog* ke dalam bentuk 255 nilai diskrit sinyal digital. Sedangkan pada ADC dengan *output digital* 16-bit akan mendapatkan resolusi ADC sebesar 65.535 nilai diskrit. Jadi, semakin tinggi bit *output* ADC, maka resolusi ADC akan semakin tinggi yang berarti ketelitian konversi sinyal *analog* ke sinyal *digital* nantinya semakin akurat.

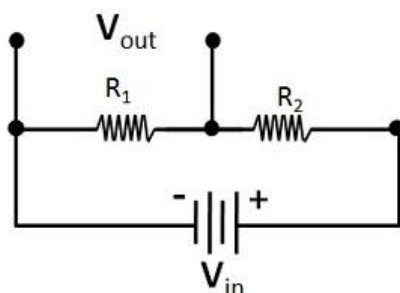
Pada mikrokontroler NodeMCU yang digunakan pada penelitian ini, ADC yang terdapat pada mikrokontroler tersebut memiliki *output* ADC sebesar 10-bit. Artinya resolusi ADC pada mikrokontroler NodeMCU sebesar 1.023 nilai diskrit. ADC pada penelitian ini digunakan untuk mengubah sinyal *analog* arus listrik yang dibaca oleh sensor arus ke dalam bentuk nilai akhir sinyal digital.

### 2.2.3 Rangkaian Pembagi Tegangan

Rangkaian pembagi tegangan (*voltage divider*) merupakan suatu rangkaian elektronika sederhana yang berfungsi untuk mengubah suatu tegangan referensi dari sumber tegangan yang lebih besar menjadi tegangan yang nilainya lebih kecil. Fungsi dari rangkaian pembagi tegangan ini adalah untuk membagi satu atau beberapa tegangan referensi menjadi beberapa tegangan *output* yang diperlukan

oleh komponen-komponen lain pada sebuah rangkaian elektronika (SparkFun Electronics, 2016).

Pada dasarnya, secara sederhana rangkaian pembagi tegangan terdiri atas dua buah resistor yang disusun secara seri, seperti pada Gambar 2.1.



**Gambar 2.1** Rangkaian Pembagi Tegangan Sederhana

Sumber : (Kho, 2016)

Aturan pada rangkaian pembagi tegangan sangat sederhana, yaitu tegangan referensi dibagi secara proporsional sesuai dengan nilai hambatan dari dua resistor yang disusun secara seri. Berikut merupakan persamaan pada rangkaian pembagi tegangan:

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2} \quad (2.1)$$

$V_{out}$  = Tegangan output

$V_{in}$  = Tegangan referensi (*input*)

$R_1$  = Resistor pertama

$R_2$  = Resistor kedua

#### 2.2.4 Sensor Arus *Current Transformer* Sensor YHDC SCT-013-000

Sensor merupakan suatu alat yang berfungsi untuk mendeteksi gejala atau sinyal yang berasal dari perubahan suatu energi seperti energi listrik, energi fisika, energi kimia, energi biologi, energi mekanik dan energi lainnya (Maulana, 2014). Sensor arus YHDC SCT-013-000 atau yang biasa disebut juga sebagai *Current Transformer sensor* merupakan sebuah sensor yang digunakan untuk mengetahui gejala atau sinyal yang terjadi pada perangkat listrik. YHDC *Current Transformer sensor* merupakan sensor yang diproduksi oleh Beijing YaoHuadechang Electronic Co., Ltd, tersedia secara luas dipasaran, dan dikenal dengan nama “*Non-invasive AC current sensor*”, sedangkan SCT-013-000 merupakan model dari sensor arus yang dimanfaatkan untuk membaca sinyal arus listrik. Sensor ini berguna untuk mengukur arus penggunaan listrik pada suatu perangkat maupun ruangan dengan cara dililitkan pada salah satu kabel perangkat listrik yang ingin dipantau penggunaannya.

Seperti pada Gambar 2.2, sensor YHDC SCT-013-000 memiliki kepala yang bisa dibuka dan ditutup, sehingga dapat dengan mudah dipasangkan pada kabel yang



bermuatan listrik atau kabel *ground* dari sebuah perangkat listrik tanpa perlu melakukan aktivitas listrik yang melibatkan arus tinggi.



**Gambar 2.2** Sensor Arus *Current Transformer Sensor* YHDC SCT-013-000

**Sumber:** (Open Energy Monitor, 2017)

Berikut ini merupakan spesifikasi dari sensor arus CT sensor YHDC SCT-013-000:

**Tabel 2.1** Spesifikasi CT Sensor YHDC SCT-013-000

Ukuran fisik	13 mm x 13 mm
Material Inti	<i>Ferrite</i>
Kekuatan Dielektrik	1000 VAC/1 min 5 mA (antara <i>shell</i> dan <i>output</i> )
Jumlah lilitan kabel	2.000 lilitan
<i>Input</i> arus listrik	0 – 100 A
Tegangan <i>output</i>	0 – 50 mV
Perbandingan arus <i>input</i> dan <i>output</i>	100 A : 0.05 A
Toleransi kerja	± 3%
Temperatur kerja	-25°C - +70°C

Sebagaimana *current transformer* pada umumnya, CT sensor YHDC ACT-013-000 memiliki tiga buah komponen yaitu lilitan utama, inti magnet, dan lilitan sekunder. Untuk mengukur penggunaan listrik pada suatu perangkat listrik atau ruangan, maka salah satu dari kabel yang bermuatan atau kabel *ground*

dimasukkan pada lubang di kepala CT sensor. Arus listrik yang terdapat pada lilitan utama menghasilkan medan magnet di inti magnet yang kemudian menginduksi listrik ke lilitan sekunder. Untuk menggunakan sensor arus tersebut dengan mikrokontroler, diperlukan beberapa perhitungan seperti perhitungan *burden resistor* dan perhitungan kalibrasi pada perangkat lunak sehingga hasil perhitungan serta pembacaan data arus listrik oleh sensor dapat lebih akurat. Adapun langkah yang dilakukan dalam proses perhitungan tersebut adalah sebagai berikut:

### 1. Hitung *primary peak-current*

$$P = \text{Arus maksimal sensor arus} \times \sqrt{2} \quad (2.2)$$

Keterangan:

$P$  = *Primary peak-current*.

### 2. Hitung *secondary peak-current*

$$S = P / n \quad (2.3)$$

Keterangan:

$S$  = *Secondary peak-current*

$n$  = Jumlah lilitan pada sensor

### 3. Hitung *burden resistor ideal*

$$I = (AREF / 2) / S \quad (2.4)$$

Keterangan:

$I$  = *Burden resistor ideal*

$AREF$  = Nilai tegangan yang digunakan pada mikrokontroler.

### 4. Hitung nilai kalibrasi

$$N = (P / S) / I \quad (2.5)$$

Keterangan:

$N$  = Nilai kalibrasi

$P$  = *primary peak-current*

$S$  = *secondary peak-current*

$I$  = *Burden resistor ideal*

## 2.2.5 Tarif Dasar Listrik

Tarif Dasar Listrik atau yang biasa disingkat dengan TDL merupakan tarif harga jual listrik yang ditetapkan oleh pemerintah untuk para pelanggan PLN. PLN merupakan satu-satunya perusahaan di Indonesia yang boleh menjual listrik secara langsung kepada masyarakat Indonesia. Maka dapat disimpulkan bahwa TDL adalah tarif untuk penggunaan listrik di Indonesia (Listrik.org, 2017).

**PENETAPAN  
PENYESUAIAN TARIF TENAGA LISTRIK (TARIFF ADJUSTMENT)**

**BULAN JULI - SEPTEMBER 2017**

NO.	GOL. TARIF	BATAS DAYA	REGULER		PRA BAYAR (Rp/kWh)
			BIAYA BEBAN (Rp/kVA/bulan)	BIAYA PEMAKAIAN (Rp/kWh) DAN BIAYA kVArh (Rp/kVArh)	
1.	R-1/TR	900 VA-RTM	*)	1.352,00	1.352,00
2.	R-1/TR	1.300 VA	*)	1.467,28	1.467,28
3.	R-1/TR	2.200 VA	*)	1.467,28	1.467,28
4.	R-2/TR	3.500 VA s.d. 5.500 VA	*)	1.467,28	1.467,28
5.	R-3/TR	6.600 VA ke atas	*)	1.467,28	1.467,28
6.	B-2/TR	6.600 VA s.d. 200 kVA	*)	1.467,28	1.467,28
7.	B-3/TM	di atas 200 kVA	**) )	Blok WBP = K x 1.035,78 Blok LWBP = 1.035,78 kVArh = 1.114,74 ****)	-
8.	I-3/TM	di atas 200 kVA	**) )	Blok WBP = K x 1.035,78 Blok LWBP = 1.035,78 kVArh = 1.114,74 ****)	-
9.	I-4/TT	30.000 kVA ke atas	*** )	Blok WBP dan Blok LWBP = 996,74 kVArh = 996,74 ****)	-
10.	P-1/TR	6.600 VA s.d. 200 kVA	*)	1.467,28	1.467,28
11.	P-2/TM	di atas 200 kVA	**) )	Blok WBP = K x 1.035,78 Blok LWBP = 1.035,78 kVArh = 1.114,74 ****)	-
12.	P-3/TR		*)	1.467,28	1.467,28
13.	L/TR, TM, TT		-	1.644,52	-

**Gambar 2.3** Daftar TDL Bulan Juli-September 2017

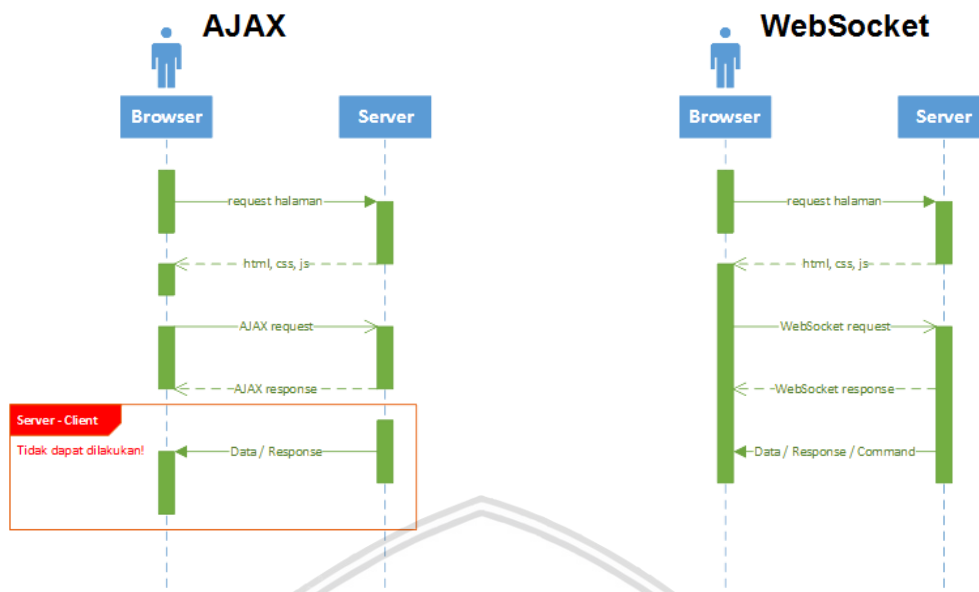
**Sumber:** (Listrik.org, 2017)

Pada Gambar 2.3 memperlihatkan ketentuan penyesuaian tarif tenaga listrik atau TDL yang dikeluarkan oleh pemerintah. Untuk kategori rumah tangga, PLN membagi TDL rumah tangga dalam beberapa golongan diantaranya 900 VA, 1.300 VA, 2.200 VA, 3.500 VA – 5500 VA, dan 6.600 VA keatas.

### 2.2.6 Protokol Websocket

Websocket merupakan sebuah protokol komunikasi *realtime* pada *web* dan aplikasi *mobile* dengan menyediakan saluran komunikasi *full-duplex* melalui satu koneksi TCP. Websocket memiliki kelebihan diantaranya penggunaan *traffic* dan *latency* jaringan yang lebih rendah dibanding dengan metode *polling* dan *long-polling* yang telah lama digunakan untuk mensimulasikan koneksi dua arah dengan cara menjaga dua koneksi tetap terhubung (Darsiwan, 2016).

Selain Websocket, para *developer* banyak menggunakan HTTP-AJAX (*Asynchronous JavaScript and XML*) untuk melakukan pembaruan konten pada sebuah halaman *web* tanpa melakukan proses *refresh* pada halaman tersebut. AJAX menggunakan komunikasi satu arah, yang artinya setiap *user* mengirim *request* ke *server*, maka *user* dapat mengirim *request* kembali setelah *server* membalas *request* sebelumnya. Jadi, *user* dan *server* perlu membuat koneksi baru setiap proses *request-reply* yang terjadi, seperti yang dijelaskan pada Gambar 2.4 (Sim., 2014).

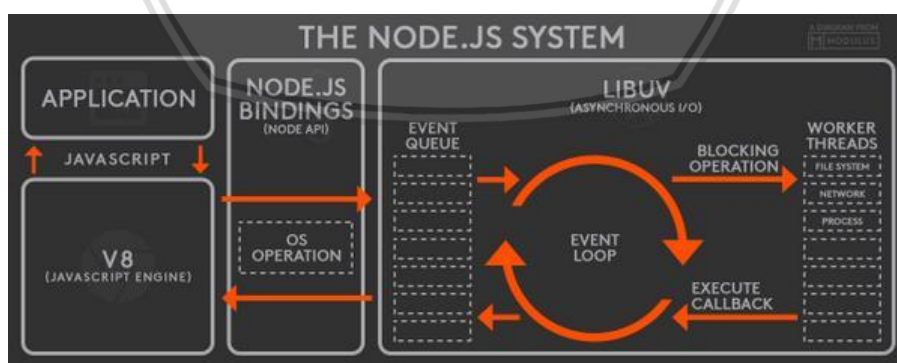


**Gambar 2.4** Perbedaan AJAX dengan Websocket

Sumber: (bertzzie.com, 2014)

## 2.2.7 Node.js

Node.js merupakan sebuah perangkat lunak yang didesain untuk pengembangan aplikasi berbasis web. Node.js ditulis menggunakan bahasa pemrograman JavaScript. Node.js pertama kali dikembangkan dan diperkenalkan oleh Ryan Dahl pada tahun 2009. Node.js dibuat untuk melengkapi peran JavaScript sehingga bisa berlaku sebagai bahasa pemrograman yang berjalan di sisi server (*server-side JavaScript*). Node.js dapat berjalan pada sistem operasi Windows, Mac OS X, dan Linux serta sudah dilengkapi dengan HTTP *server library* yang memungkinkan untuk menjalankan *web server* tanpa menggunakan program *web server* seperti Apache atau Nginx (Chhetri, 2016).



**Gambar 2.5** Arsitektur dari Node.js

Sumber: (Sagar, 2016)

Berbeda dengan bahasa pemrograman di sisi server (*server-side application*) pada umumnya yang bersifat *blocking*, Node.js ini bersifat *non-blocking*, seperti halnya JavaScript bekerja. Maksud dari *blocking* disini sederhananya adalah,

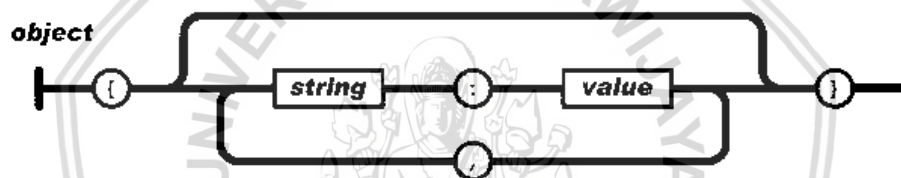


bahwa suatu kode program akan dijalankan dari awal hingga selesai sebelum dapat beralih ke kode program berikutnya. Node.js bekerja dengan sistem *event-driven* dan menggunakan JavaScript *engine* dari Google yang bernama V8. V8 *engine* juga digunakan oleh Google pada browser Google Chrome milik mereka. Arsitektur sederhana dari Node.js dapat dilihat pada Gambar 2.5 (Lutfi, 2017).

### 2.2.8 JSON

JSON merupakan singkatan dari *JavaScript Object Notation* adalah sebuah format untuk berbagi data yang merupakan turunan dari bahasa pemrograman JavaScript. Akan tetapi, format JSON tersedia untuk bahasa pemrograman lain seperti Java, PHP, Python, dan Ruby.

JSON menggunakan ekstensi file *.json* ketika berdiri sendiri. Namun saat JSON didefinisikan pada format bahasa pemrograman lain, json tampil dengan menggunakan tanda petik sebagai *JSON string*. Objek JSON ditulis dalam bentuk pasangan *key-value* yang diawali oleh kurung kurawal, memiliki tanda titik dua diantara *key* dan *value*, serta setiap *key-value* dipisahkan oleh tanda koma seperti pada Gambar 2.6.



Gambar 2.6 Format Penulisan Objek JSON

Sumber: (Shin, 2007)

### 2.2.9 OpenEnergyMonitor

*OpenEnergyMonitor* merupakan sebuah *open source project* yang dikembangkan untuk membantu memantau penggunaan energi khususnya energi listrik. *Project* ini awalnya dikembangkan oleh Trystan Lea dan Glyn Hudson yang menyediakan informasi mengenai alat pemantau penggunaan listrik berbasis mikrokontroler mulai dari rancangan rangkaian elektronika dan *library* pemrograman yang berguna untuk membantu para *developer* dalam mengembangkan sistem pemantau penggunaan energi listrik (*OpenEnergyMonitor*, 2016).

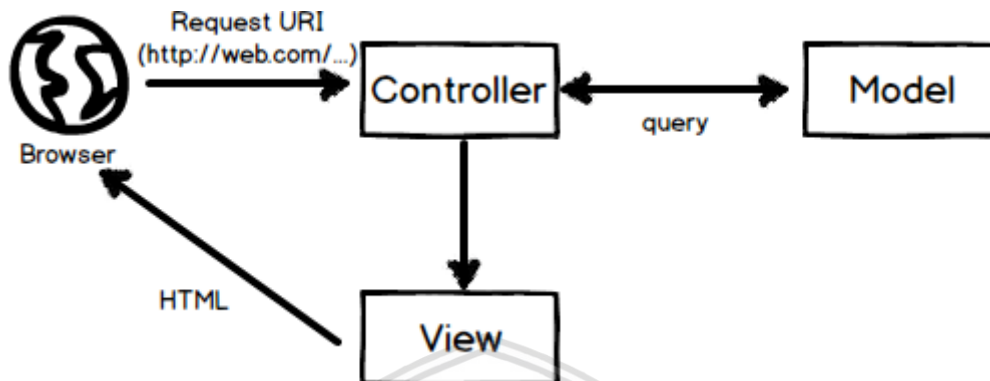
### 2.2.10 Framework PHP CodeIgniter

CodeIgniter merupakan sebuah aplikasi *web* bersifat *open source* yang digunakan untuk membangun aplikasi PHP secara dinamis. CodeIgniter menjadi sebuah *framework* PHP yang dibangun dengan menggunakan konsep MVC (*Model, View, Controller*) untuk mengembangkan sebuah *website* dinamis dengan menggunakan bahasa pemrograman PHP (Cloud Hosting Indonesia, PT., 2017).

Konsep MVC sendiri menggunakan prinsip SoC (*Separation of Concern*) dimana kode-kode program dibagi menjadi beberapa bagian, dimana ada kode program



yang berhubungan dengan basis data (*Model*), ada kode program yang berhubungan dengan tampilan ke pengguna (*View*), dan ada kode program yang menghubungkan antara *Model* dan *View* (*Controller*) seperti yang digambarkan pada Gambar 2.7 (Sim, 2013).



**Gambar 2.7** Arsitektur Sederhana dari Konsep MVC

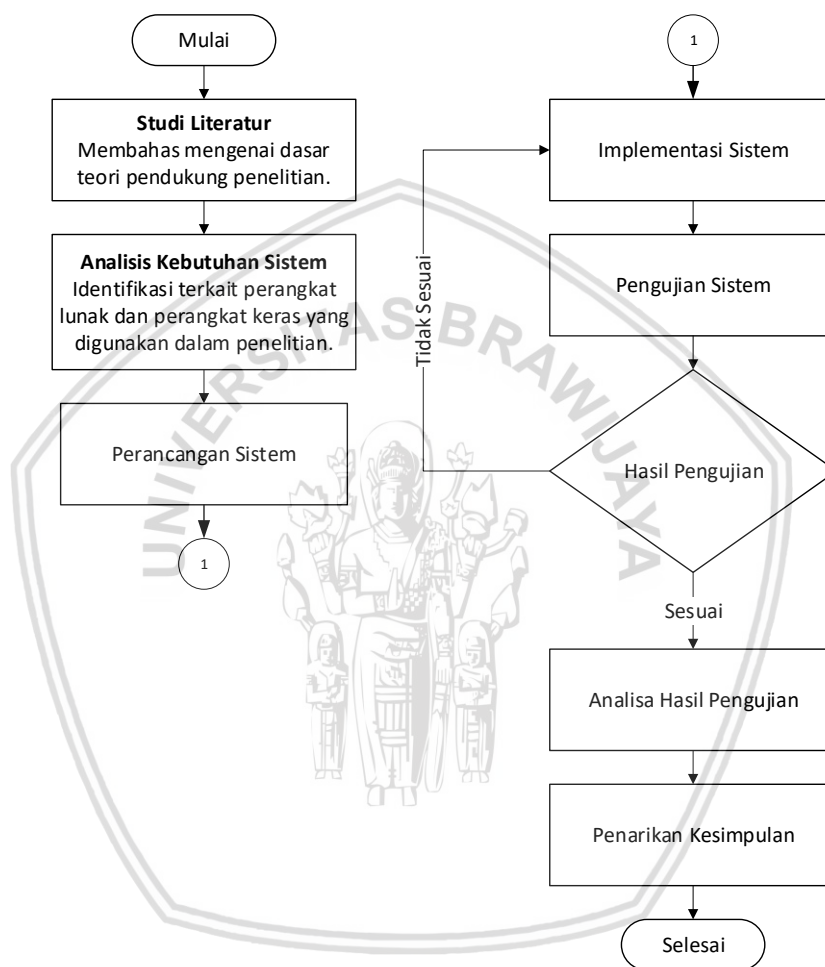
Sumber: (bertzzie.com, 2017)



## BAB 3 METODOLOGI

### 3.1 Metodologi Penelitian

Pada sub bab ini akan dijelaskan mengenai langkah-langkah yang akan dilakukan dalam melaksanakan penelitian. Adapun langkah-langkah dari penelitian yang akan dilakukan dapat dilihat pada diagram alir berikut:



**Gambar 3.1** Diagram Alir Metode Penelitian

### 3.2 Studi Literatur

Studi literatur menjelaskan mengenai dasar teori yang digunakan dalam mendukung implementasi sistem yang dibangun pada penelitian. Adapun studi literatur yang digunakan dalam menunjang perancangan sistem terdiri dari:

1. Mikrokontroler NodeMCU dengan pemrograman menggunakan bahasa C.
2. Sensor arus Current Transformer YHDC SCT-013-000.
3. Protokol komunikasi Websocket berbasis node.js JavaScript.
4. Pemrograman *web* berbasis *framework* PHP CodeIgniter.

Diharapkan dari studi literatur tersebut dapat memecahkan masalah yang ada pada proses pembuatan sistem, sehingga sistem yang dibangun dapat lebih tepat dan terarah sesuai rencana.

### 3.3 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan untuk mengetahui apa saja yang dibutuhkan dalam melaksanakan penelitian ini. Analisa kebutuhan sistem terbagi menjadi dua bagian, yaitu kebutuhan perangkat keras dan kebutuhan perangkat lunak.

#### 3.3.1 Kebutuhan Perangkat Keras

Perangkat keras yang diperlukan dalam menunjang penelitian ini diantaranya:

1. Mikrokontroler NodeMCU sebagai mikrokontroler yang digunakan untuk pengolahan data dari sensor arus listrik.
2. Sensor arus *Current Transformer* YHDC SCT-013-000 yang digunakan untuk mengukur arus listrik AC (*Alternating Current*) pada terminal listrik, dimana data arus listrik tersebut akan digunakan untuk mengukur penggunaan daya listrik yang digunakan.
3. Terminal Listrik yang digunakan oleh peralatan listrik untuk memperoleh energi listrik agar dapat beroperasi.
4. Laptop yang digunakan untuk mengakses antarmuka *web* sistem *monitoring* daya listrik, membuat perangkat lunak sistem, serta berfungsi sebagai *Websocket server*, *web server*, dan *database server*.
5. Alat *power meter* yang digunakan untuk membandingkan hasil pengukuran arus, tegangan, dan daya listrik dari perangkat *monitoring* daya listrik yang dibangun.

#### 3.3.2 Kebutuhan Perangkat Lunak

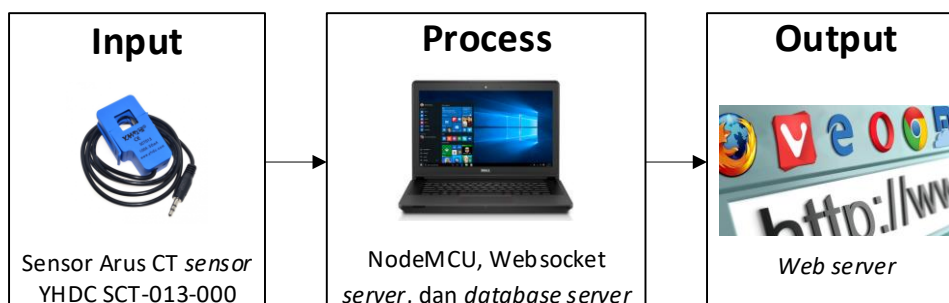
Perangkat lunak yang digunakan dalam menunjang penelitian ini diantaranya:

1. Arduino IDE dengan Bahasa C sebagai bahasa pemrograman yang digunakan pada mikrokontroler NodeMCU.
2. Sublime Text dengan HTML, PHP, dan JavaScript sebagai bahasa pemrograman yang digunakan pada *Websocket server*, *web server*, dan *database server*.
3. Node.js sebagai *environment* dari *Websocket server* yang dibangun.
4. XAMPP dengan Apache sebagai *web server* dan MySQL sebagai *database server*.

### 3.4 Perancangan Sistem

Perancangan sistem dilakukan agar nantinya sistem dapat memenuhi kebutuhan fungsional seperti yang diharapkan. Teori-teori, referensi, serta sumber informasi lain digabungkan dengan ilmu yang telah dimiliki kemudian

diimplementasikan untuk merancang dan mengembangkan sistem *monitoring* daya listrik. Adapun diagram blok sederhana dari sistem *monitoring* daya listrik yang dibangun adalah sebagai berikut:



**Gambar 3.2** Diagram Blok Perancangan *Input*, *Proses*, dan *Output*

Pada bagian *input* terdapat sensor arus CT sensor YHDC SCT-013-000. Sensor arus tersebut membaca sinyal *analog* arus listrik yang mengalir pada terminal listrik. Pada bagian *process* hasil pembacaan sinyal *analog* arus tersebut nantinya akan diproses oleh mikrokontroler NodeMCU yang selanjutnya akan dikirimkan ke Websocket server menggunakan protokol komunikasi Websocket. Websocket server bertindak sebagai perantara antara database server dengan perangkat keras NodeMCU. Database server berfungsi sebagai tempat untuk menyimpan data hasil *monitoring* yang dikirimkan oleh mikrokontroler NodeMCU. Pada bagian *output* terdapat web server yang akan menampilkan data *monitoring* dalam antarmuka web dan sebagai media interaksi antara pengguna dengan sistem *monitoring* daya listrik.

### 3.5 Implementasi Sistem

Setelah perancangan sistem dilakukan, maka proses penelitian akan berlanjut pada tahap implementasi sistem, dimana pada proses implementasi sistem peneliti akan mengimplementasikan desain dan perancangan yang telah dilakukan menjadi sebuah kesatuan sistem yang utuh. Pada proses implementasi sistem terdapat beberapa fungsi yang harus berjalan sesuai dengan rancangan sistem sebelumnya, diantaranya:

1. Perangkat keras mampu melakukan akuisisi data *monitoring* nilai arus listrik yang dibaca oleh sensor arus serta mengirimkan data *monitoring* tersebut ke Websocket server.
2. Websocket server mampu berkomunikasi data *monitoring* listrik dengan perangkat keras dan menyimpan data *monitoring* listrik ke database server.
3. Web server mampu menampilkan data *monitoring* penggunaan listrik dari setiap perangkat *monitoring* dan mengatur konfigurasi dari perangkat *monitoring* listrik tersebut.

### 3.6 Pengujian Sistem

Pada proses pengujian sistem dilakukan dengan acuan pada perancangan sistem, diantaranya:

1. Pengujian perangkat keras, yaitu dengan menguji data *monitoring* yang dilakukan oleh perangkat keras berupa arus, tegangan, dan daya listrik dengan menggunakan alat Taff Energy Power Meter – DEM1499.
2. Pengujian performa penerimaan data *monitoring*, yaitu untuk menguji berapa waktu yang diperlukan oleh sistem untuk setiap data *monitoring* listrik yang didapat.
3. Pengujian perangkat lunak, yaitu dengan melakukan pengujian fungsionalitas dari *web monitoring* daya listrik yang dibangun.

### 3.7 Kesimpulan dan Saran

Tahap akhir dari penelitian adalah penarikan kesimpulan dan saran. Kesimpulan diambil sebagai jawaban atas rumusan masalah yang telah dijabarkan pada BAB 1, yaitu:

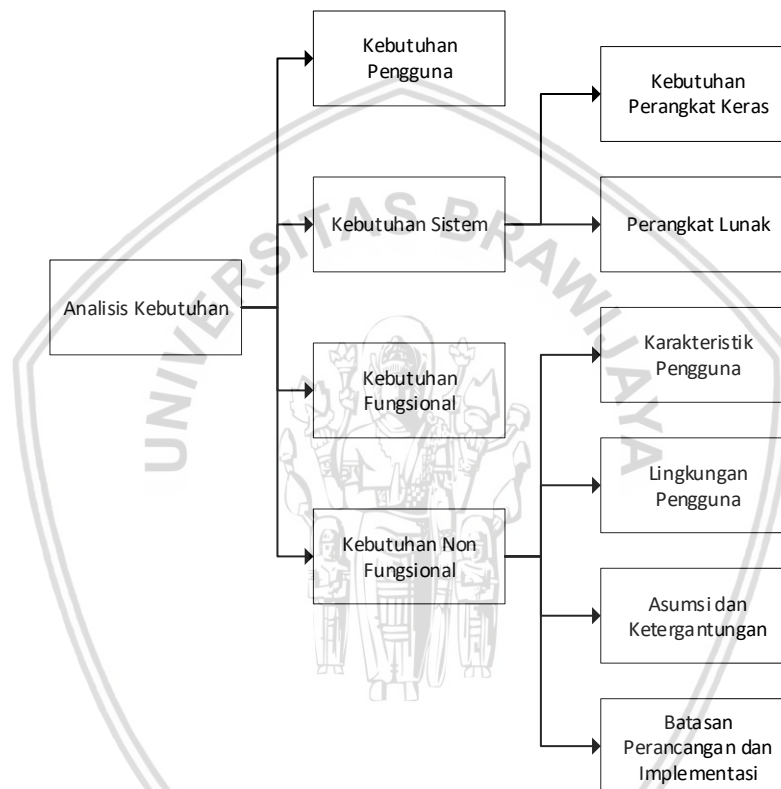
1. Bagaimana akurasi dari perangkat keras *monitoring* daya listrik yang dibuat dalam melakukan *monitoring* penggunaan listrik di tiap ruangan yang ada pada sebuah rumah?
2. Bagaimana performa penerimaan data *monitoring* listrik pada sistem *monitoring* daya listrik yang dibangun?
3. Bagaimana kinerja fungsionalitas dari *web monitoring* daya listrik yang dibangun?

Sedangkan saran memiliki tujuan untuk memperbaiki kesalahan atau kekurangan serta sebagai acuan dalam penyempurnaan atau pengembangan penelitian selanjutnya.



## BAB 4 ANALISIS KEBUTUHAN

Pada bab ini akan membahas beberapa kebutuhan yang diperlukan untuk melakukan penelitian diantaranya kebutuhan pengguna, kebutuhan sistem, kebutuhan fungsional, dan kebutuhan non fungsional. Pada kebutuhan sistem dibagi menjadi dua bagian, diantaranya kebutuhan perangkat keras dan kebutuhan perangkat lunak. Sedangkan kebutuhan non fungsional akan dibagi menjadi 4 bagian, diantaranya adalah karakteristik pengguna, lingkungan pengguna, asumsi dan ketergantungan, serta batasan perancangan dan implementasi seperti pada Gambar 4.1.



Gambar 4.1 Diagram Analisa Kebutuhan

### 4.1 Kebutuhan Pengguna

Pada kebutuhan pengguna akan menjelaskan gambaran bagaimana pengguna akan berinteraksi dengan sistem yang dibuat, agar sistem dapat bekerja sesuai dengan tujuan yang diharapkan. Pengguna berinteraksi dengan sistem melalui *web* yang akan menampilkan hasil *monitoring listrik* yang didapat dari perangkat. Hasil *monitoring listrik* bergantung pada penggunaan perangkat listrik yang digunakan pada setiap perangkat *monitoring*.

## 4.2 Kebutuhan Sistem

### 4.2.1 Kebutuhan Perangkat Keras

Perangkat keras yang akan digunakan dalam proses implementasi pada penelitian ini adalah sebagai berikut.

- a. *Laptop* digunakan sebagai *Websocket server*, *database server*, *web server*, serta untuk membuat kode program dan untuk mengakses halaman *web monitoring listrik*. Pada penelitian ini peneliti menggunakan *laptop* Dell Inspiron 14 7447 Pandora karena spesifikasi dari *laptop* tersebut sudah sangat mumpuni dijadikan sebagai *server*. Bentuk fisik dari *laptop* tersebut seperti pada Gambar 4.2.



**Gambar 4.2** *Laptop* Dell Inspiron 14 7447 Pandora

Spesifikasi dari *laptop* diatas adalah sebagai berikut:

<i>Laptop</i>	: Dell Inspiron 14 7447 Pandora
Prosesor	: Intel® Core™ i7-4720HQ @2.6 GHz, turbo up to 3.6 GHz
Memori RAM	: 8 GB DDR3
Penyimpanan	: Patriot Spark 256 GB SSD
Kartu Grafis	: NVIDIA GeForce GTX 850M 4 GB DDR3
Layar	: 14 inci dengan resolusi 1366x768

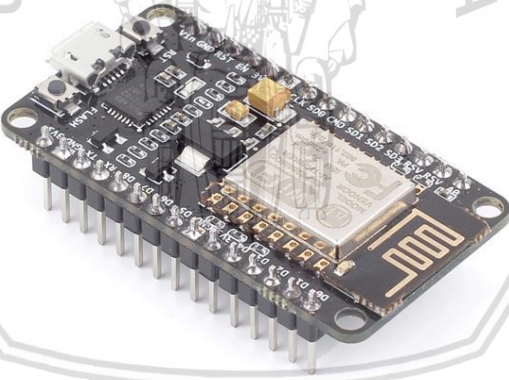
- b. *Wireless Router* atau *Access Point* digunakan sebagai media penghubung antara *node monitoring listrik* dengan *server*. Peneliti menggunakan *wireless router* Netgear DGN1000 karena pada penelitian ini menggunakan media komunikasi berbasis Wi-Fi antara perangkat *monitoring* dengan *server* yang mana *wireless router* tersebut mendukungnya. Bentuk fisik dari *Wireless*

router Netgear DGN1000 yang digunakan seperti yang diperlihatkan pada Gambar 4.3.



**Gambar 4.3** Wireless router Netgear DGN1000

- c. Mikrokontroler NodeMCU seperti pada Gambar 4.4 digunakan sebagai mikrokontroler untuk perangkat *monitoring listrik*. Mikrokontroler NodeMCU digunakan karena bentuk fisiknya yang ringkas, mendukung komunikasi Wi-Fi, dan memiliki sebuah *pin analog* yang digunakan untuk menghubungkan sensor *monitoring* arus listrik. Nantinya NodeMCU akan mengolah data arus listrik yang dibaca oleh sensor arus dan mengirimkan data *monitoring* arus listrik ke *Websocket server*.



**Gambar 4.4** Mikrokontroler NodeMCU

Pada Tabel 4.1 dibawah akan dijelaskan mengenai spesifikasi dari mikrokontroler NodeMCU:

**Tabel 4.1** Spesifikasi Mikrokontroler NodeMCU

Parameter	Nilai
Protokol Wi-Fi	IEEE 802.11 b/g/n
Rentang Frekuensi Kerja	2,412 – 2,484 GHz
Kekuatan Pancaran Sinyal	802.11b : +16 ± 2 dBm (di 11 Mbps) 802.11g : +14 ± 2 dBm (di 54 Mbps)

	802.11n : $+13 \pm 2$ dBm (di HT20, MCS7)
Sensitivitas Penerimaan Sinyal	802.11b : -93 dBm (di 11 Mbps, CCK) 802.11g : -85 dBm (di 54 Mbps, OFDM) 802.11n : -82 dBm (di HT20, MCS7)
Jenis Antena	Antena <i>on-board</i>
Kapabilitas IO	UART, I2C, PWM, GPIO, 1 ADC 10-bit
Karakteristik Operasi	3.3 V ketika beroperasi 15 mA arus <i>output</i> tiap pin GPIO 12-200 mA arus ketika beroperasi <200 uA ketika <i>standby</i>
Temperatur Kerja	-40 - 125°C
Kecepatan Serial	110-921600 bps
Mode Kerja Wi-Fi	Station (STA), Access Point (AP), dan Station + Access Point (STA + AP)
Mode Keamanan	WEP/WPA-PSK/WPA2-PSK
Mode Enkripsi	WEP64/WEP128/TKIP/AES
<i>Firmware Upgrade</i>	<i>Local Serial Port</i> atau <i>OTA Remote Upgrade</i>
Protokol Jaringan	IPv4, TCP/UDP/FTP/HTTP

- d. Terminal listrik yang pada penelitian ini dibutuhkan sebagai tempat untuk meletakkan sensor *monitoring* arus listrik. Ada sedikit pelindung kabel terminal listrik yang dikelupas untuk meletakkan sensor arus listrik. Nantinya perangkat listrik yang akan dipantau penggunaan listriknya akan dihubungkan pada terminal listrik tersebut.
- e. Sensor *monitoring* arus listrik CT sensor YHDC SCT-013-000 digunakan untuk membaca nilai arus listrik yang mengalir pada terminal listrik. Sensor YHDC SCT-013-000 pada penelitian ini digunakan karena kemampuannya membaca nilai arus listrik hingga 100 A dan kemudahan dalam menggunakannya yang hanya dengan meletakkan sensor tersebut pada salah satu bagian kabel dari perangkat yang akan dibaca nilai arus listriknya. Nilai pembacaan arus listrik tersebut nantinya akan diolah oleh mikrokontroler NodeMCU yang selanjutnya akan dikirim ke Websocket server dan disimpan pada database server.
- f. Alat Taff *Energy Power Meter* seperti pada Gambar 4.5 yang pada penelitian ini digunakan untuk membandingkan hasil pengukuran arus dan daya listrik dari perangkat *monitoring* yang dibuat. Taff *Energy Power Meter* memiliki fitur untuk membaca penggunaan tegangan, arus, dan daya listrik dari perangkat listrik yang terpasang.



**Gambar 4.5** Taff Energy Power Meter

Spesifikasi dari Alat Taff Energy Power Meter yang digunakan seperti yang dijelaskan pada Tabel 4.2 dibawah ini:

**Tabel 4.2** Spesifikasi Taff Energy Power Meter

Tegangan Input/Output	230 VAC/50 Hz
Arus Maksimal	16A
Tegangan Kerja	190-270 VAC
Kisaran Daya	1-3680 Watt
Frekuensi Kerja	46-65 Hz
Baterai	3 V (2x baterai tipe LR44/AG13)

#### 4.2.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang akan digunakan dalam proses implementasi pada penelitian ini adalah sebagai berikut.

- Sistem operasi dari laptop *server* dan laptop pengguna adalah Microsoft Windows 10 Home 64-bit. Windows 10 digunakan karena bersifat *user friendly* dan umum digunakan oleh pengguna.
- Node.js digunakan sebagai *Websocket server* dengan bahasa pemrograman JavaScript. Node.js digunakan karena pada Node.js sudah terdapat modul Websocket yang dibutuhkan pada penelitian ini tanpa perlu memasang modul Websocket secara terpisah.
- XAMPP digunakan sebagai *web server* berbasis Apache dan *database server* berbasis MySQL *database*. XAMPP digunakan karena bersifat *cross-platform* dan tidak perlu memasang *web server* dan *database server* secara terpisah.
- Arduino IDE yang digunakan untuk membuat kode program dan nantinya kode program tersebut akan ditanamkan pada mikrokontroler NodeMCU. Arduino IDE digunakan karena mendukung pemrograman pada mikrokontroler NodeMCU dengan bahasa pemrograman C.



- e. Sublime Text yang pada penelitian ini digunakan untuk membuat kode program pada *Websocket server*, *web server*, dan *database server*.
- f. *Library* *emonlib* pada penelitian ini digunakan pada mikrokontroler NodeMCU dalam melakukan perhitungan atau konversi sinyal *analog* yang dibaca oleh sensor ke dalam bentuk nilai arus listrik.
- g. *Library* *Websocket* yang digunakan pada mikrokontroler NodeMCU dan *node.js* untuk komunikasi data antara perangkat *monitoring* listrik dengan *server*.
- h. *Library* *ArduinoJson* digunakan untuk menyimpan data *monitoring* listrik yang dilakukan oleh perangkat keras. Data yang disimpan berupa data dalam bentuk objek *json*.

### 4.3 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan dimana sistem dapat memberikan layanan atau fungsi yang sesuai dengan tujuan dibentuknya sistem tersebut kepada pengguna. Kebutuhan fungsional yang dibutuhkan pada penelitian ini diantaranya:

1. Sensor arus listrik mampu melakukan akuisisi data arus listrik dari terminal listrik pada sebuah ruangan yang dipantau penggunaannya dalam bentuk sinyal *analog*.
2. Mikrokontroler NodeMCU mampu mengubah sinyal *analog* arus listrik yang dibaca oleh sensor arus listrik ke dalam satuan arus listrik (ampere), lalu mengirimkan hasil pembacaan arus listrik tersebut beserta *MAC address* dari NodeMCU ke *Websocket server*.
3. *Websocket server* mampu menerima data arus listrik yang dikirimkan oleh perangkat *monitoring* listrik dan melakukan kalkulasi penggunaan daya listrik dari data arus listrik yang diterima. Selanjutnya *Websocket server* akan mengirimkan data tersebut ke *database server*.
4. *Database server* mampu menyimpan data *monitoring* listrik, daftar perangkat *monitoring* listrik dan daftar ruangan yang dipantau, serta daftar tarif dasar listrik.
5. *Web monitoring* listrik mampu menampilkan data penggunaan listrik (Watt dan KWh) pada terminal listrik yang ada pada ruangan dalam satuan waktu tertentu dan menampilkan estimasi biaya penggunaan listrik pada satu bulan terakhir dengan cara menghitung KWh penggunaan listrik dengan nilai TDL. *Web monitoring* listrik juga mampu mendaftar dan menghapus daftar perangkat *monitoring listrik* serta menambah, menghapus, mengubah, dan memilih nilai TDL pada daftar TDL.

## 4.4 Kebutuhan Non Fungsional

### 4.4.1 Karakteristik Pengguna

Implementasi sistem menggunakan perangkat *monitoring listrik* yang bekerja secara aktif melakukan *monitoring listrik* setiap detik dan data hasil *monitoring* akan dikirimkan ke Websocket *server* yang selanjutnya akan disimpan di *database server*.

### 4.4.2 Lingkungan Pengguna

Lingkungan pengguna berada di tiap-tiap ruangan pada sebuah rumah dengan tiga kamar. Perangkat *monitoring listrik* masing-masing diletakkan pada terminal listrik yang terdapat di tiap kamar. Selanjutnya, pengujian perangkat juga akan dilakukan di tiap kamar tersebut.

### 4.4.3 Batasan Desain Sistem

Sistem yang diimplementasikan memiliki beberapa batasan desain sistem untuk mencapai tujuan yang diharapkan, yaitu:

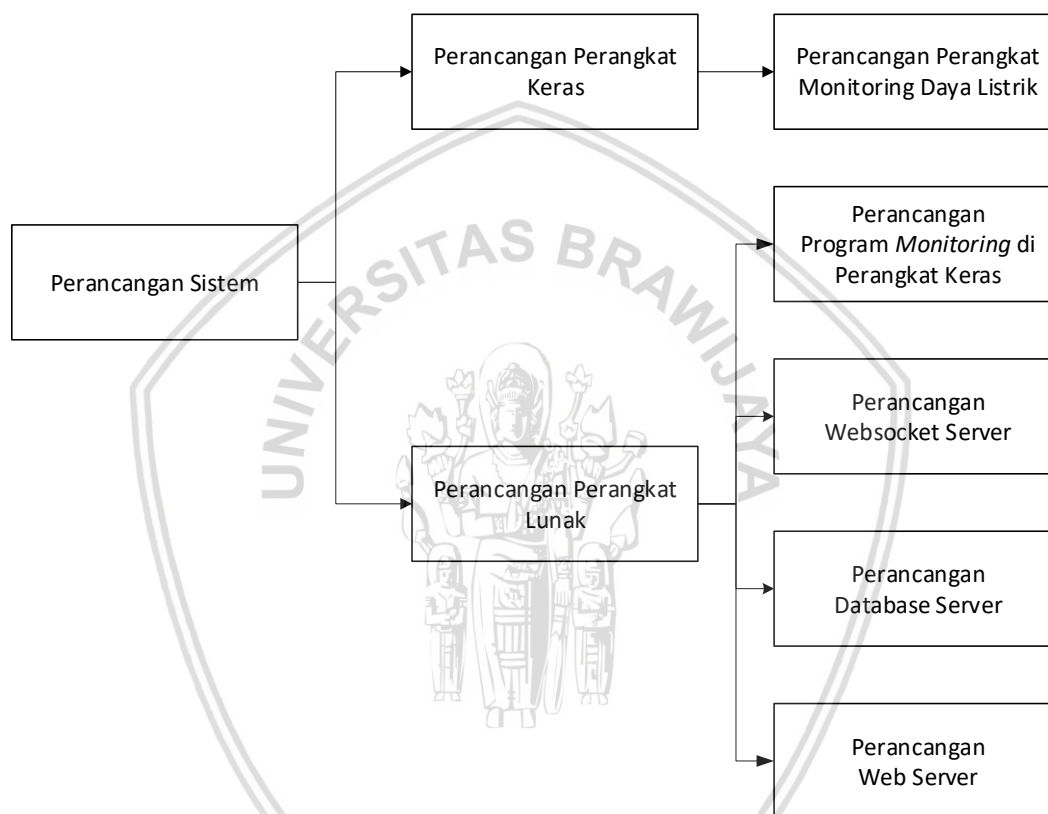
1. Perangkat *monitoring listrik* diletakkan pada setiap terminal listrik yang ada di ruangan pada sebuah rumah.
2. Fitur dari sistem *monitoring listrik* diantaranya mampu menambahkan dan menghapus perangkat *monitoring listrik* dan data ruangan; mampu menghitung penggunaan daya listrik, total penggunaan daya listrik, serta estimasi biaya penggunaan listrik; dan sistem mampu mengubah daftar nilai TDL serta memilih TDL aktif.
3. Data *monitoring listrik* disimpan pada *database server* dan *output monitoring* akan ditampilkan di *web*.

## BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan menjelaskan tentang perancangan dan implementasi dari sistem yang dibangun secara keseluruhan.

### 5.1 Perancangan Sistem

Pada tahap ini dijelaskan perancangan sistem secara umum. Perancangan sistem terdiri dari perancangan perangkat keras dan perangkat lunak. Berikut merupakan susunan dari perangkat sistem yang dibangun.



**Gambar 5.1** Struktur Perancangan Sistem

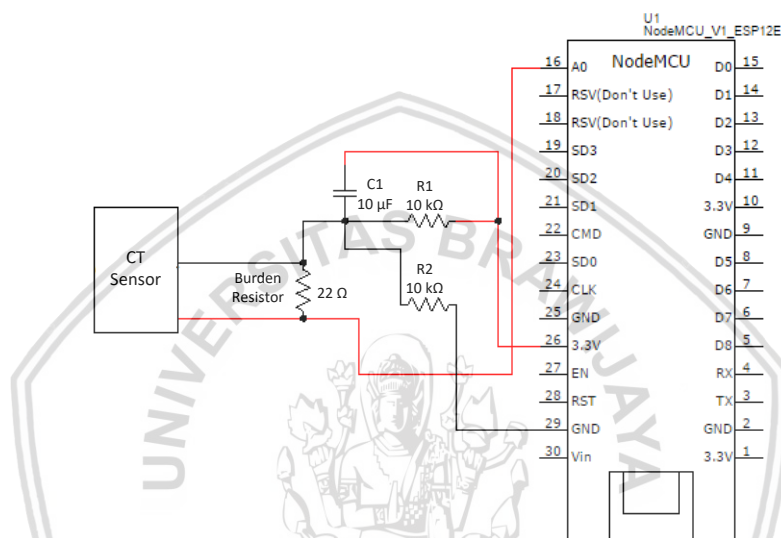
Berdasarkan Gambar 5.1, struktur perancangan sistem terbagi menjadi dua bagian, yaitu perancangan perangkat keras dan perancangan perangkat lunak. Perancangan perangkat keras merupakan perancangan perangkat (*device*) yang digunakan untuk melakukan *monitoring listrik*. Sedangkan perancangan perangkat lunak meliputi perancangan program *monitoring* di NodeMCU, Websocket *server* yang digunakan untuk komunikasi dari perangkat *monitoring* ke *web server* serta *database server*, perancangan *database server* yang digunakan untuk menyimpan data hasil *monitoring listrik* yang dilakukan oleh perangkat keras, serta *web server* yang digunakan sebagai antarmuka interaksi pengguna dengan sistem *monitoring* daya listrik.

### 5.1.1 Perancangan Perangkat Keras

Pada bagian ini akan dijelaskan mengenai perancangan dari perangkat keras sistem yang dibangun. Mulai dari penentuan komponen elektronik apa saja yang akan digunakan, perancangan komponen-komponen tersebut menjadi sebuah kesatuan sistem, hingga pembuatan perangkat jadi.

#### 5.1.1.1 Perancangan Perangkat *Monitoring* Daya Listrik

Pada Gambar 5.2 dibawah ini merupakan rancangan skematik rangkaian dari perangkat keras *monitoring* listrik yang digunakan.



**Gambar 5.2** Skematik Perangkat Keras

Adapun beberapa komponen yang digunakan pada skematik diatas adalah sebagai berikut:

- Mikrokontroler NodeMCU
- Sensor arus listrik CT sensor YHDC SCT-013-000
- Resistor 22  $\Omega$  sebagai *burden* resistor
- 2 x 10 k $\Omega$  resistor untuk pembagi tegangan 3,3 V menjadi 1,65 V
- 10  $\mu$ F Kapasitor

Komponen elektronika seperti resistor dan kapasitor diperlukan dalam menghubungkan sensor arus listrik dengan mikrokontroler NodeMCU, karena *output* sinyal *analog* dari sensor arus harus dikondisikan terlebih dahulu sehingga cocok dengan syarat *input analog* dari mikrokontroler NodeMCU.

Resistor R1 dan R2 pada rangkaian diatas berfungsi sebagai pembagi tegangan yang mengubah tegangan referensi 3,3 V dari mikrokontroler NodeMCU untuk menghasilkan *output* tegangan sebesar 1,65 V sesuai persamaan 2.1. Kapasitor C1 memiliki nilai reaktansi yang rendah hanya beberapa ratus ohm, sehingga kapasitor dengan ukuran 10  $\mu$ F sudah cukup.

Sensor arus listrik CT sensor YHDC SCT-013-000, memiliki keluaran sinyal *analog* arus listrik yang perlu diubah sesuai dengan *burden* resistor yang digunakan. Adapun proses pemilihan *burden* resistor dan nilai kalibrasi sensor yang digunakan pada perangkat *monitoring* listrik didapat dengan langkah-langkah sebagai berikut:

### 1. Karakteristik arus listrik pada CT sensor

Sensor arus listrik CT sensor YHDC SCT-013-000 memiliki rentang pengukuran arus dari 0 A hingga maksimal 100 A.

### 2. Hitung *primary peak-current*

Dengan menggunakan persamaan 2.1, maka:

$$P = \text{Nilai maksimal arus} \times \sqrt{2}$$

$$P = 100 \text{ A} \times 1,414$$

$$P = 141,4 \text{ A}$$

### 3. Hitung *secondary peak-current*

Sensor arus listrik YHDC SCT-013-000 memiliki jumlah lilitan sebanyak 2000 lilitan. Dengan menggunakan persamaan 2.2, maka didapatkan:

$$S = P / n$$

$$S = 141,4 \text{ A} / 2000$$

$$S = 0,0707 \text{ A}$$

### 4. Hitung *burden* resistor ideal

Mikrokontroler NodeMCU memiliki tegangan Vcc sebesar 3.3 V. Dengan menggunakan persamaan 2.3, maka *burden* resistor idealnya adalah sebagai berikut:

$$I = (AREF / 2) / S$$

$$I = (3,3 \text{ V} / 2) / 0,0707 \text{ A}$$

$$I = 1,65 \text{ V} / 0,0707 \text{ A}$$

$$I = 23,3 \Omega$$

### 5. Hitung nilai kalibrasi

Untuk dapat melakukan pengukuran arus listrik menggunakan sensor arus YHDC SCT-013-000 diperlukan nilai kalibrasi sehingga hasil pengukuran dapat dilakukan dengan lebih akurat. Dengan menggunakan persamaan 2.4, maka proses perhitungan kalibrasi adalah sebagai berikut:

$$N = (P / S) / I$$

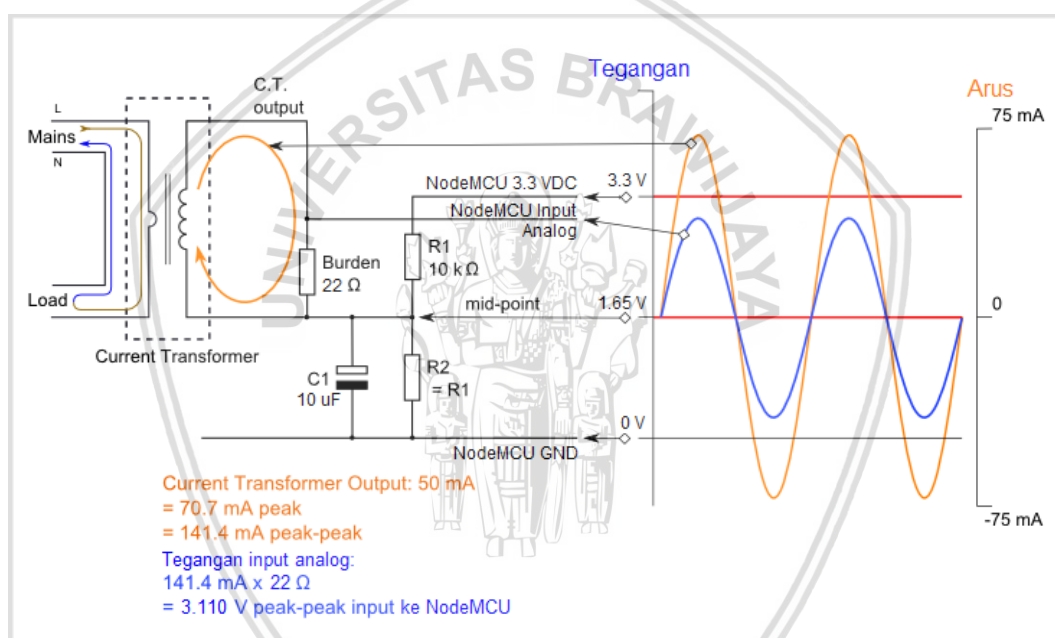
$$N = (141,4 \text{ A} / 0,0707 \text{ A}) / 22 \Omega$$

$$N = 2000 / 22 \Omega = 90,9$$



Dikarenakan  $23,3 \Omega$  bukanlah nilai resistor yang umum, maka diambil nilai resistor umum yang terdekat, yaitu  $22 \Omega$ .

Komponen-komponen yang terpasang pada mikrokontroler NodeMCU seperti skematik perangkat keras pada Gambar 5.2 dialiri oleh tegangan 3,3 V. Pada *input* pin A0 dihubungkan dengan *burden* resistor lalu bagian positif dari sensor arus listrik. Kemudian pada bagian kapasitor dan bagian negatif dari sensor arus listrik dihubungkan pada resistor pembagi tegangan dan bagian *ground* di NodeMCU. Sensor arus listrik dihubungkan pada pin *analog* A0 dikarenakan sensor arus listrik merupakan sensor *analog* yang artinya sensor membaca sinyal arus listrik dalam bentuk sinyal *analog*. Nantinya sinyal *analog* yang dibaca oleh sensor arus akan dikondisikan ke dalam bentuk sinyal *analog* yang cocok dengan *input analog* 3,3 V pada mikrokontroler NodeMCU, seperti pada Gambar 5.3. Setelah itu, maka sinyal analog yang telah dikondisikan akan diubah ke dalam bentuk nilai digital oleh program perangkat lunak.



**Gambar 5.3** Pengkondisian Sinyal *Analog* Listrik oleh Sensor Arus

**Sumber:** (Open Energy Monitor, 2017)

Rangkaian perangkat keras *monitoring* listrik berguna untuk mengukur arus listrik pada terminal listrik yang akan diukur. Untuk nilai tegangan yang dibaca, digunakan nilai tegangan tetap sebesar 220 V yang merupakan standar tegangan rumah di Indonesia.

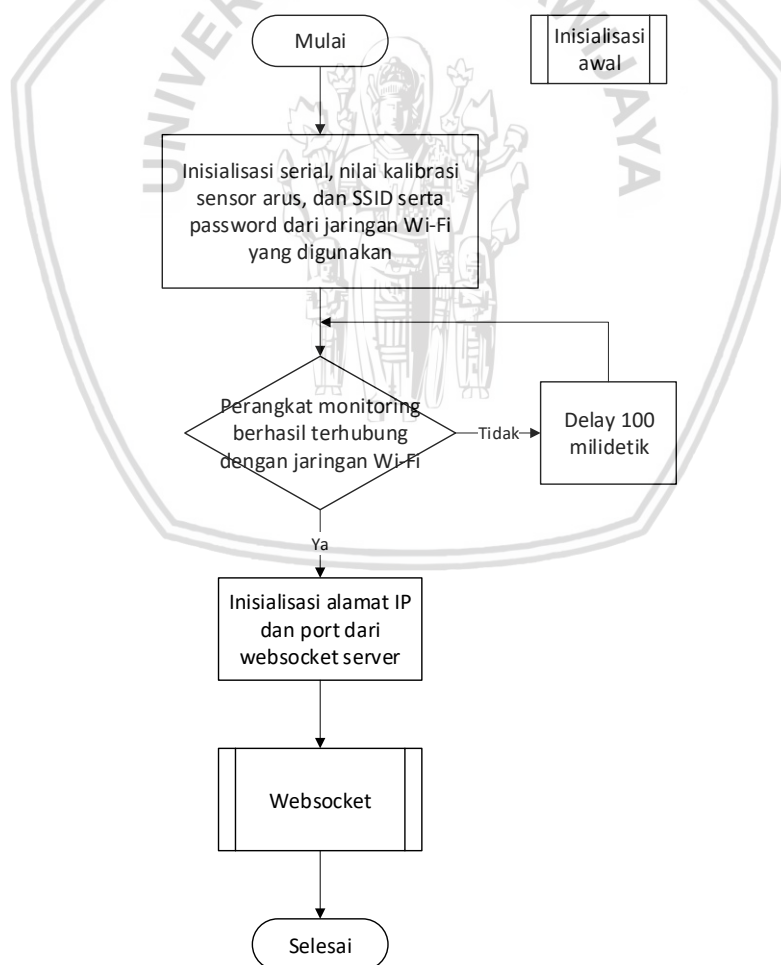
### 5.1.2 Perancangan Perangkat Lunak

Pada tahap perancangan perangkat lunak akan dijelaskan mengenai dengan alur kerja dari kode sumber yang nantinya akan menjadi antarmuka bagi setiap pengguna yang memakai sistem tersebut.

### 5.1.2.1 Perancangan Program *Monitoring* di Perangkat Keras

Perancangan program *monitoring* di perangkat keras dilakukan agar perangkat keras mampu mendapatkan nilai arus listrik yang dibaca oleh sensor arus dan data *monitoring* dapat dikirim ke Websocket server. Data yang dikirimkan oleh perangkat keras diantaranya MAC Address dari NodeMCU perangkat keras yang digunakan dan data nilai pembacaan arus listrik dari sensor arus. Berikut merupakan *flowchart* dari program *monitoring* di perangkat keras yang dibagi menjadi beberapa kategori diantaranya inisialisasi awal (meliputi inisialisasi *baud rate serial*, nilai kalibrasi dari sensor arus, jaringan Wi-Fi yang digunakan, dan alamat IP serta port dari Websocket server), Websocket (meliputi kondisi ketika perangkat *monitoring* terhubung dengan Websocket server, ketika perangkat *monitoring* mendapat pesan/request dari Websocket server, dan ketika koneksi antara perangkat *monitoring* dengan Websocket server terputus), dan baca arus (meliputi pembuatan variabel json yang terdiri dari MAC Address NodeMCU perangkat *monitoring*, data nilai pembacaan arus listrik yang dilakukan oleh sensor, dan pengiriman data tersebut ke Websocket server).

#### 1. Inisialisasi awal

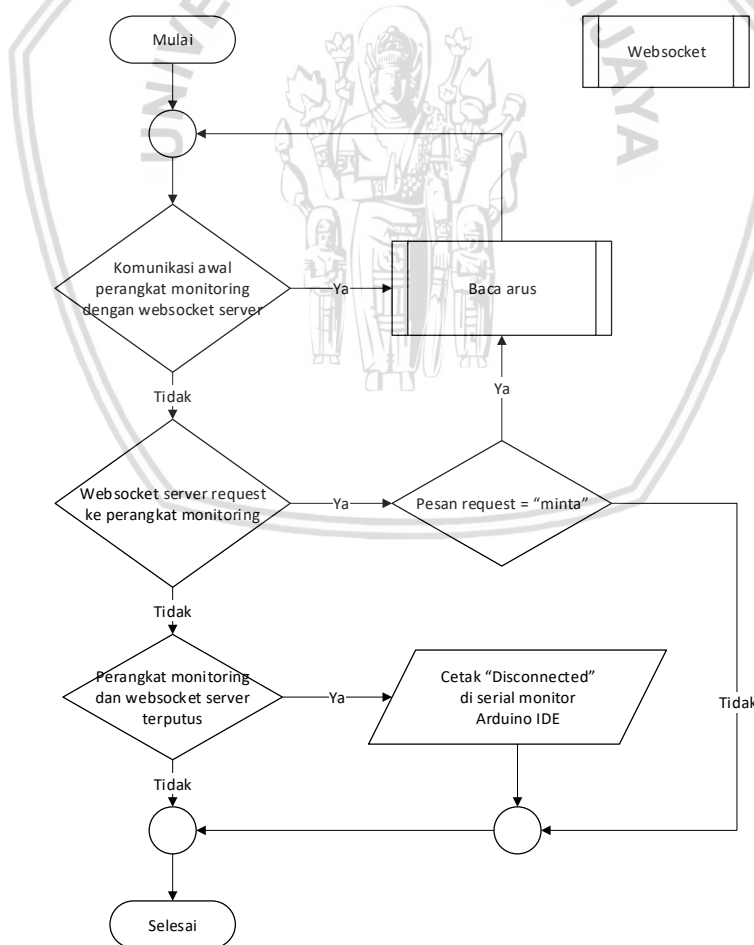


**Gambar 5.4** *Flowchart* proses inisialisasi awal

Berdasarkan Gambar 5.4, dapat dijelaskan alur kerja program *monitoring* di NodeMCU pada kategori inisialisasi awal adalah sebagai berikut:

- Pada saat perangkat *monitoring* diaktifkan, maka akan dilakukan inisialisasi *baud rate serial*, nilai kalibrasi sensor arus, serta SSID dan *password* dari jaringan Wi-Fi yang digunakan.
- Perangkat *monitoring* mencoba terhubung dengan *router/access point* pada jaringan Wi-Fi yang sudah diatur sebelumnya. Jika perangkat *monitoring* belum terhubung dengan jaringan Wi-Fi, maka akan ada *delay* selama 100 milidetik sebelum mencoba kembali proses hubungan dengan jaringan Wi-Fi tersebut. Proses tersebut akan berjalan hingga perangkat *monitoring* berhasil terhubung dengan jaringan Wi-Fi.
- Setelah perangkat *monitoring* berhasil terhubung dengan jaringan Wi-Fi, maka dilakukan proses inisialisasi alamat IP dan *port* dari Websocket server yang dituju. Setelah itu, akan dilakukan proses komunikasi Websocket antara perangkat *monitoring* dengan Websocket server yang lebih jelasnya akan dijelaskan pada kategori Websocket.

## 2. Websocket

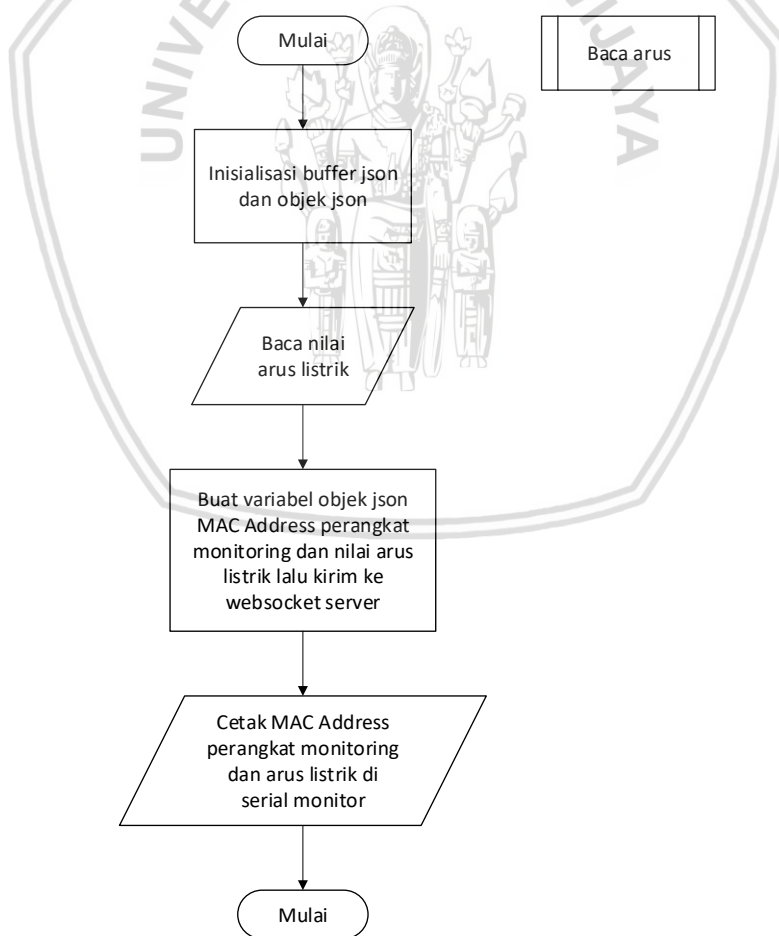


**Gambar 5.5** Flowchart Websocket

Berdasarkan Gambar 5.5, *flowchart* Websocket merupakan bagian dimana terjadi proses komunikasi Websocket antara perangkat *monitoring* dan Websocket server. Berikut merupakan penjelasan dari *flowchart* pada Gambar 5.5:

- Setelah inialisasi awal dilakukan, maka selanjutnya adalah proses menjalin komunikasi awal Websocket antara perangkat *monitoring* dengan Websocket server. Ketika koneksi awal telah terjalin, maka perangkat *monitoring* akan menjalankan fungsi baca arus yang mana fungsi baca arus akan dijelaskan lebih lanjut pada kategori baca arus.
- Ketika koneksi awal telah terjalin dan perangkat *monitoring* mendapat *request* dari Websocket server, maka perangkat *monitoring* akan mengecek isi *request* yang diterima. Apabila isi *request* adalah "minta", maka perangkat *monitoring* akan menjalankan fungsi baca arus. Selain itu, maka *request* akan diabaikan.
- Apabila koneksi Websocket antara perangkat *monitoring* dan Websocket server terputus, maka di *serial monitor* Arduino IDE akan mencetak pesan "Disconnected".

### 3. Baca arus



Gambar 5.6 *Flowchart* baca arus

Berdasarkan Gambar 5.6, *flowchart* baca arus merupakan bagian dimana perangkat *monitoring* melakukan pembacaan nilai arus yang didapat dari sensor. Berikut merupakan penjelasan dari *flowchart* pada Gambar 5.6:

- Pembacaan arus dimulai dari inisialisasi *buffer* dan objek json dengan nama "data". Karena data yang dikirimkan dari perangkat *monitoring* ke Websocket *server* nantinya akan dikirimkan menjadi satu paket data dengan format json.
- Setelah *buffer* dan objek json dibuat, maka akan dilakukan pembacaan nilai arus listrik pada terminal listrik yang didapat dari sensor arus.
- Setelah nilai arus listrik didapat, maka nilai arus Irms tersebut akan disimpan pada objek json dengan *key* "value" dan MAC Address dari perangkat *monitoring* tersebut pada objek json dengan *key* "id".
- Setelah objek json dibuat, maka objek json tersebut akan dijadikan satu paket pada variabel "databuf" dengan tipe data *StreamString* dan dikirimkan ke Websocket *server*.
- Setelah data tersebut dikirim, maka pada *serial monitor* Arduino IDE akan mencetak MAC Address dari perangkat *monitoring* tersebut dan nilai pembacaan arus listrik yang didapat.

#### 5.1.2.2 Perancangan Websocket Server

Websocket server bertugas untuk mengirimkan *request* data arus listrik ke perangkat *monitoring* listrik dan meneruskan data yang diterima dari perangkat *monitoring* ke *database server*. Di Websocket *server* terdapat mekanisme bagaimana Websocket *server* bertindak sebagai perantara antara perangkat *monitoring* dengan *database server* dan mengolah data arus listrik yang diterima dari perangkat *monitoring*. Berikut merupakan *flowchart* dari Websocket *server*.

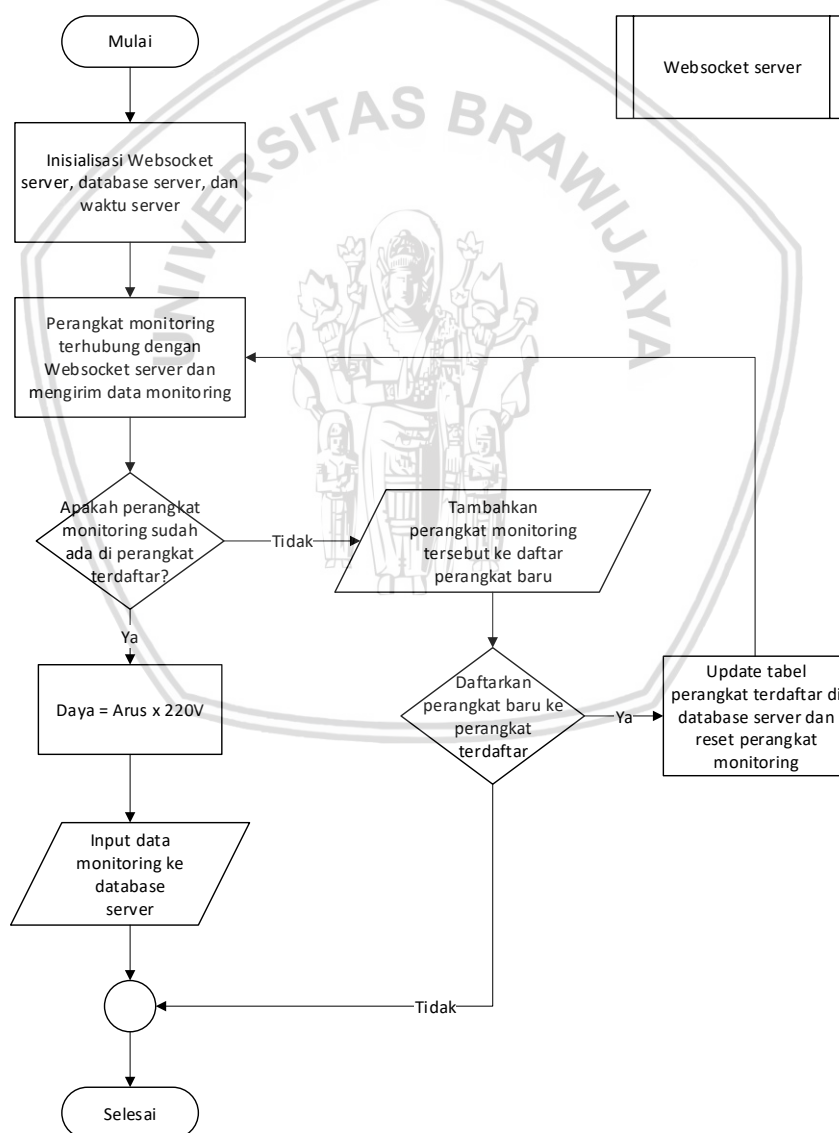
Berdasarkan Gambar 5.7, *flowchart* komunikasi antara perangkat *monitoring* listrik dengan Websocket *server* merupakan proses pertukaran data yang terjadi antara perangkat *monitoring* dengan Websocket *server*. Berikut merupakan penjelasan dari *flowchart* pada Gambar 5.7:

- Pada saat Websocket *server* pertama kali dijalankan, maka Websocket *server* akan melakukan inisialisasi alamat IP dan *port* yang digunakan oleh Websocket *server*, konfigurasi target *database server* yang digunakan, serta waktu *server* yang nantinya digunakan untuk proses *input* data *monitoring* listrik ke *database server*.
- Ketika ada perangkat *monitoring* listrik yang mengirimkan data *monitoring* ke Websocket *server*, maka Websocket *server* akan melakukan *query* ke tabel perangkat terdaftar di *database server* dan melakukan pengecekan apakah perangkat *monitoring* tersebut sudah ada di tabel perangkat terdaftar atau tidak. Jika perangkat *monitoring* listrik tersebut tidak ditemukan dalam tabel perangkat terdaftar, maka MAC *address* dari perangkat *monitoring* listrik tersebut akan dimasukkan pada tabel perangkat baru di *database server*. Selanjutnya, perangkat baru tersebut akan muncul di kategori perangkat baru



dan pengguna dapat memilih untuk mendaftarkan perangkat baru tersebut atau tidak. Jika pengguna ingin mendaftarkan perangkat baru, maka pengguna harus memilih perangkat baru tersebut dan memilih ruangan dimana perangkat itu diletakkan. Setelah itu, maka tabel perangkat baru dan perangkat terdaftar di *database server* akan diperbarui dan perangkat yang sebelumnya didaftarkan harus di-reset terlebih dahulu agar dapat mengirim data *monitoring* ke Websocket server.

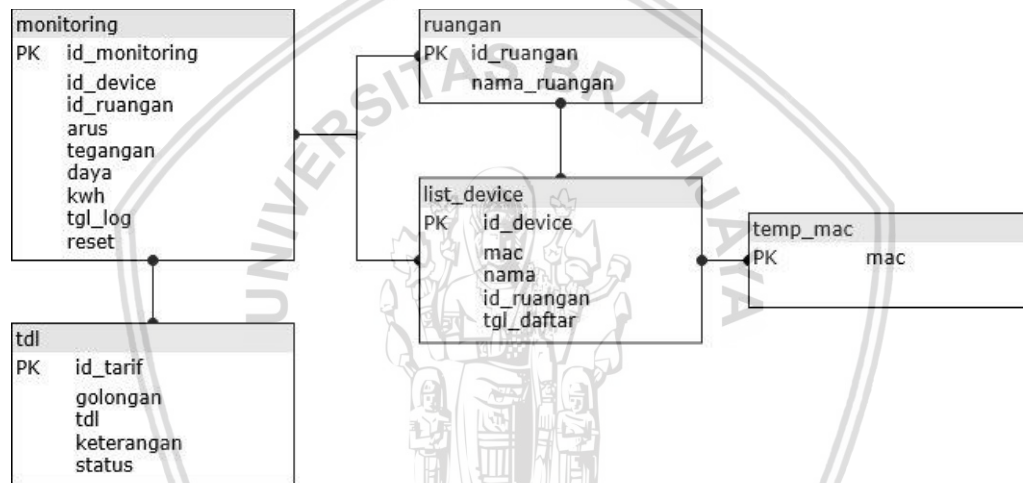
- c. Apabila perangkat *monitoring* listrik yang terhubung dengan Websocket server sudah ada dalam perangkat terdaftar, maka Websocket server akan menjalankan perhitungan daya listrik dimana data arus listrik yang diterima dikalikan dengan nilai tegangan sebesar 220 V dan selanjutnya data hasil perhitungan tersebut akan dimasukkan ke dalam tabel *monitoring* di *database server*.



**Gambar 5.7** Flowchart Websocket server

### 5.1.2.3 Perancangan Database Server

Database server dirancang agar data arus listrik yang dikirimkan dari perangkat keras dapat disimpan dan nantinya dapat diakses oleh web server. Data monitoring tersebut disimpan dalam bentuk tabel yang dibagi menjadi 5 kelompok yaitu, tabel *monitoring*, tabel perangkat terdaftar, tabel ruangan, tabel TDL, dan tabel perangkat baru. Peneliti dapat membuat desain relasi hubungan antar tabel sehingga nantinya dapat diketahui hubungan antar tabel dalam database tersebut. Relasi antar tabel tersebut didapat dari perancangan tabel pada database yang sudah peneliti jelaskan diatas. Relasi didapat dengan terdapatnya nama kolom yang sama atau saling berkaitan satu dengan lainnya. Desain dari relasi tersebut nantinya dapat memudahkan untuk membaca tabel-tabel yang ada di database, sehingga peneliti dapat melakukan implementasi dalam pembuatan database serta query database dengan lebih mudah. Adapun desain dari database yang dirancang seperti pada Gambar 5.8.



Gambar 5.8 Desain Relasi Database

Berikut merupakan penjelasan dari desain relasi database pada Gambar 5.8:

#### 1. Tabel Monitoring

Tabel *monitoring* merupakan tabel pada database yang menampung data seperti informasi dari perangkat monitoring listrik, ruangan dimana perangkat monitoring listrik tersebut dipasang, besarnya arus listrik yang dibaca oleh sensor arus, dan sebagainya. Detail dari tabel *monitoring* seperti pada Tabel 5.1 dibawah ini:

Tabel 5.1 Tabel monitoring

Nama Kolom	Tipe Data dan Nilai
<i>id_monitoring</i>	bigint(255) auto increment primary key
<i>id_device</i>	int(100) foreign key tabel list_device
<i>id_ruangan</i>	int(100) foreign key tabel ruangan
<i>arus</i>	float

tegangan	int(100)
daya	float
kwh	float
tgl_log	datetime
reset	int(1)

Berikut merupakan penjelasan dari setiap kolom yang ada pada tabel *monitoring*.

- Id\_monitoring* yang merupakan *primary key* pada tabel *monitoring* dengan tipe data *bigint(255)* digunakan untuk menampung ID dari setiap data *monitoring* yang masuk ke *database server*. *Id\_monitoring* bersifat *auto increment* yang artinya *Id\_monitoring* akan otomatis terisi angka secara urut dimulai dari angka atau ID 1.
- Id\_device* yang merupakan *foreign key* mengacu pada kolom *id\_device* pada tabel *list\_device* dengan tipe data *int(100)* berisi *id\_device* dari perangkat *monitoring* listrik yang digunakan. Penjelasan lebih lanjut mengenai kolom *id\_device* akan dijelaskan pada tabel *list\_device*.
- Id\_ruangan* yang merupakan *foreign key* mengacu pada kolom *id\_ruangan* pada tabel *ruangan* dengan tipe data *int(100)* menyimpan *id\_ruangan* dari nama ruangan dimana perangkat *monitoring* listrik dipasang. *Id\_ruangan* akan dijelaskan lebih rinci pada tabel *ruangan*.
- Arus dengan tipe data *float* pada tabel *monitoring* ini menyimpan nilai arus listrik yang dibaca oleh sensor listrik.
- Tegangan dengan tipe data *int(100)* berisi besarnya nilai tegangan listrik yang selanjutnya akan digunakan bersama dengan data arus listrik untuk mendapatkan nilai daya listrik. Tegangan pada penelitian ini secara *default* diatur pada nilai 220 V.
- Daya dengan tipe data *float* menyimpan nilai hasil perkalian antara arus listrik dengan tegangan listrik per satuan waktu tertentu.
- Kwh dengan tipe data *float* menyimpan data jumlah penggunaan daya listrik dalam satu jam.
- Tgl\_log* dengan tipe data *datetime* pada penelitian ini digunakan untuk menampung detail waktu dari *monitoring* yang dilakukan oleh perangkat *monitoring* listrik.
- Reset dengan tipe data *int(1)* digunakan untuk membedakan data kwh pada bulan terakhir dengan bulan sebelumnya. Untuk data kwh pada bulan terakhir akan memiliki nilai reset 1 dan bulan sebelumnya akan memiliki nilai reset 0. Reset digunakan karena estimasi biaya penggunaan listrik yang ditampilkan di *web monitoring* listrik nantinya hanya menampilkan estimasi biaya

penggunaan listrik pada bulan terakhir yang berarti akan kembali ke nol pada setiap awal bulan waktu 00:00:00.

## 2. Tabel Ruangan

Tabel ruangan pada database ini berisi informasi mengenai ruangan dimana perangkat *monitoring* listrik dipasang. Pada tabel ruangan berisi nama ruangan dan ID dari ruangan tersebut yang detailnya akan dijelaskan sebagai berikut:

**Tabel 5.2** Tabel Ruangan

Nama Kolom	Tipe Data dan Nilai
id_ruangan	int(100) <i>auto increment primary key</i>
nama_ruangan	varchar(128)

Dibawah ini merupakan penjelasan dari kolom-kolom pada Tabel 5.2:

- Id\_ruangan sebagai *primary key* pada tabel ruangan memiliki tipe data int(100) menyimpan ID dari nama ruangan dimana perangkat *monitoring* listrik dipasang. Id\_ruangan bersifat *auto increment* yang berarti nilai id\_ruangan akan dimulai dari 1 dan secara otomatis akan bertambah jika ada info ruangan baru yang ditambahkan ke *database*.
- Nama\_ruangan dengan tipe data varchar(128) digunakan untuk menyimpan nama ruangan dimana perangkat *monitoring* listrik ditempatkan.

## 3. Tabel list\_device

Tabel list\_device menyimpan informasi terkait perangkat *monitoring* listrik yang digunakan. Berikut merupakan detail dari kolom list\_device pada *database*:

**Tabel 5.3** Tabel list\_device

Nama Kolom	Tipe Data dan Nilai
id_device	int(100) <i>auto increment primary key</i>
mac	varchar(128)
nama	varchar(100)
id_ruangan	int(100) <i>foreign key</i> tabel ruangan
tgl_daftar	datetime

Berikut merupakan penjelasan dari kolom-kolom pada Tabel 5.3:

- Id\_device dengan tipe data int(100) yang merupakan *primary key* pada tabel list\_device menyimpan informasi ID dari perangkat *monitoring* listrik yang digunakan. Pada id\_device bersifat *auto increment* yang mana nilai pada id\_device akan bertambah secara otomatis setiap ada perangkat *monitoring* listrik yang ditambahkan. Id\_device dimulai dari ID 1.

- b. Mac dengan tipe data varchar(128) menyimpan informasi MAC *address* dari mikrokontroler NodeMCU yang digunakan pada perangkat keras *monitoring* listrik.
- c. Nama dengan tipe data varchar(100) digunakan untuk menyimpan nama dari perangkat *monitoring* listrik yang digunakan agar memudahkan pengguna dalam membedakan antara satu perangkat *monitoring* listrik dengan perangkat *monitoring* listrik lainnya.
- d. Id\_ruangan dengan tipe data int(100) *foreign key* dengan tabel ruangan menyimpan ID ruangan dimana perangkat *monitoring* listrik ditempatkan.
- e. Tgl\_daftar dengan tipe data datetime menyimpan informasi mengenai waktu perangkat *monitoring* listrik didaftarkan pada *database*.

#### 4. Tabel temp\_mac

Tabel temp\_mac menyimpan daftar informasi MAC *address* dari perangkat *monitoring* listrik baru yang terdeteksi oleh *server*. Pada tabel temp\_mac hanya memiliki satu kolom yaitu kolom mac dengan tipe data varchar(100).

#### 5. Tabel TDL

Tabel TDL pada penelitian ini digunakan untuk menyimpan daftar TDL yang digunakan untuk perhitungan estimasi biaya penggunaan listrik. Pada tabel tdl terdapat informasi mengenai golongan TDL, tarif listrik per KWh, dan lainnya yang mana akan dijelaskan sebagai berikut:

**Tabel 5.4** Tabel TDL

Nama Kolom	Tipe Data dan Nilai
id_tarif	int(100) <i>auto increment primary key</i>
golongan	int(11)
tdl	float
keterangan	varchar(100)
status	int(1)

Berikut merupakan penjelasan dari kolom-kolom pada Tabel 5.4:

- a. Id\_tarif yang merupakan *primary key* dari tabel tdl memiliki tipe data int(100) menyimpan informasi ID tdl dalam *database*. Id\_tarif bersifat *auto increment* yang berarti ID\_tarif akan bertambah satu setiap ada data TDL baru yang ditambahkan. Id\_tarif dimulai dari ID 1.
- b. Golongan dengan tipe data int(11) yang pada penelitian ini menyimpan informasi mengenai golongan TDL seperti contoh 900 VA, 1300 VA, 2200 VA, dan sebagainya. Golongan TDL ini mengacu pada ketentuan tarif dasar listrik yang dikeluarkan oleh PT. PLN Persero.



- c. TDL dengan tipe data float menyimpan informasi dari biaya per KWh dari tiap golongan TDL, yang mana biaya per KWh tersebut juga mengacu pada ketentuan tarif dasar listrik yang dikeluarkan oleh PT. PLN Persero.
- d. Keterangan yang menyimpan informasi apakah golongan TDL tersebut termasuk ke dalam golongan TDL bersubsidi atau non-subsidi. Keterangan memiliki tipe data varchar(100).
- e. Status yang pada penelitian ini digunakan untuk perhitungan estimasi biaya penggunaan listrik dari setiap total KWh perangkat *monitoring* listrik. Jadi, estimasi biaya penggunaan listrik nantinya akan dihitung berdasarkan golongan TDL yang dipilih oleh pengguna. Untuk golongan TDL yang dipilih maka akan memiliki nilai status 1, sedangkan golongan TDL lainnya akan memiliki nilai status 0. Status memiliki tipe data int(1).

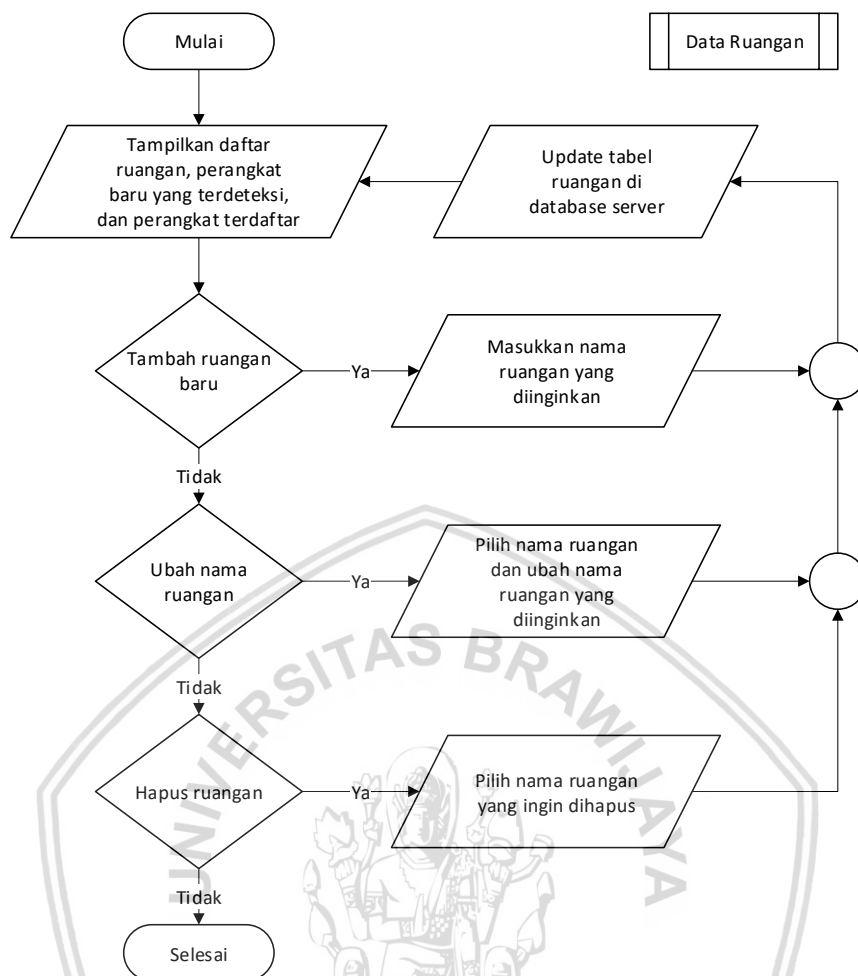
#### 5.1.2.4 Perancangan Web Monitoring Daya Listrik

Perancangan *web monitoring* daya listrik dilakukan agar memudahkan peneliti dalam mengimplementasikan beberapa hal dalam *web* tersebut, seperti penambahan data ruangan yang dipantau, penambahan perangkat *monitoring listrik* yang digunakan, menampilkan data *monitoring listrik*, serta pengubahan daftar TDL. Berikut merupakan *flowchart* dari antarmuka *web monitoring listrik* yang dibagi menjadi beberapa kategori diantaranya, data ruangan, perangkat baru dan perangkat terdaftar, *monitoring*, serta tarif dasar listrik.

##### 1. Data Ruangan

Seperti pada Gambar 5.9, maka penjelasan mengenai alur kerja pada bagian ruangan di halaman data ruangan adalah sebagai berikut:

- a. Pada saat *web monitoring* pertama kali diakses, maka akan diarahkan pada halaman data ruangan, dimana pada halaman data ruangan tersebut akan ditampilkan informasi daftar nama ruangan, daftar perangkat baru yang terdeteksi, dan perangkat *monitoring listrik* yang terdaftar.
- b. Apabila pengguna ingin menambahkan ruangan baru, maka pengguna harus memasukkan nama ruangan yang ingin ditambahkan. Setelah itu, ruangan baru tersebut akan ditambahkan pada tabel ruangan di *database server*.
- c. Apabila pengguna ingin mengubah nama ruangan yang sudah ada, maka pengguna harus memilih ruangan yang ingin diubah namanya, lalu pengguna memasukkan nama ruangan baru yang dimaksud. Setelah itu, tabel ruangan di *database server* akan diperbarui.
- d. Apabila pengguna ingin menghapus ruangan yang sudah ada, maka pengguna harus memilih ruangan yang ingin dihapus. Setelah itu, ruangan tersebut akan dihapus dari tabel ruangan di *database server*.
- e. Setelah itu, maka akan kembali ke tampilan halaman data ruangan.



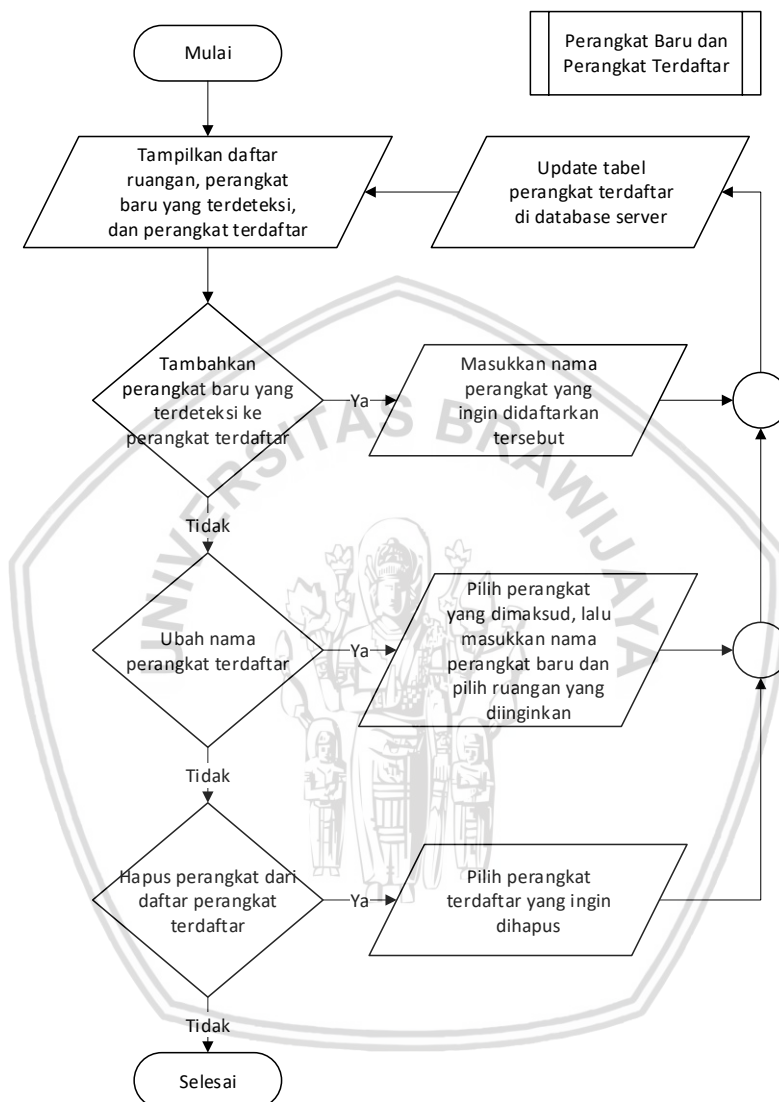
Gambar 5.9 Flowchart Data Ruangan

## 2. Perangkat Baru dan Perangkat Terdaftar

Seperti pada Gambar 5.10, maka penjelasan mengenai alur kerja pada bagian perangkat baru dan perangkat terdaftar di halaman data ruangan adalah sebagai berikut:

- Pada saat di halaman data ruangan dan ada perangkat *monitoring* baru terdeteksi, maka MAC *address* dari perangkat *monitoring* tersebut akan dimasukkan pada tabel perangkat baru di *database server* dan ditampilkan pada bagian perangkat baru di halaman data ruangan. Selanjutnya, pengguna dapat menambahkan perangkat baru tersebut ke perangkat terdaftar. Pengguna harus memasukkan nama perangkat tersebut dengan nama yang diinginkan. Setelah itu, maka data perangkat baru tersebut akan berpindah dari tabel perangkat baru ke tabel perangkat terdaftar di *database server*.
- Apabila pengguna ingin mengubah nama perangkat terdaftar, maka pengguna harus memilih perangkat terdaftar yang dimaksud dan selanjutnya memasukkan nama perangkat yang diinginkan serta memilih ruangan dimana perangkat tersebut berada. Setelah itu, maka tabel perangkat terdaftar di *database server* akan diperbarui.

- c. Apabila pengguna ingin menghapus salah satu perangkat terdaftar, maka pengguna harus memilih perangkat yang ingin dihapus. Setelah itu, perangkat tersebut akan dihapus dari daftar perangkat terdaftar pada tabel perangkat terdaftar di *database server*.
- d. Setelah itu, maka akan kembali ke tampilan halaman perangkat terdaftar.



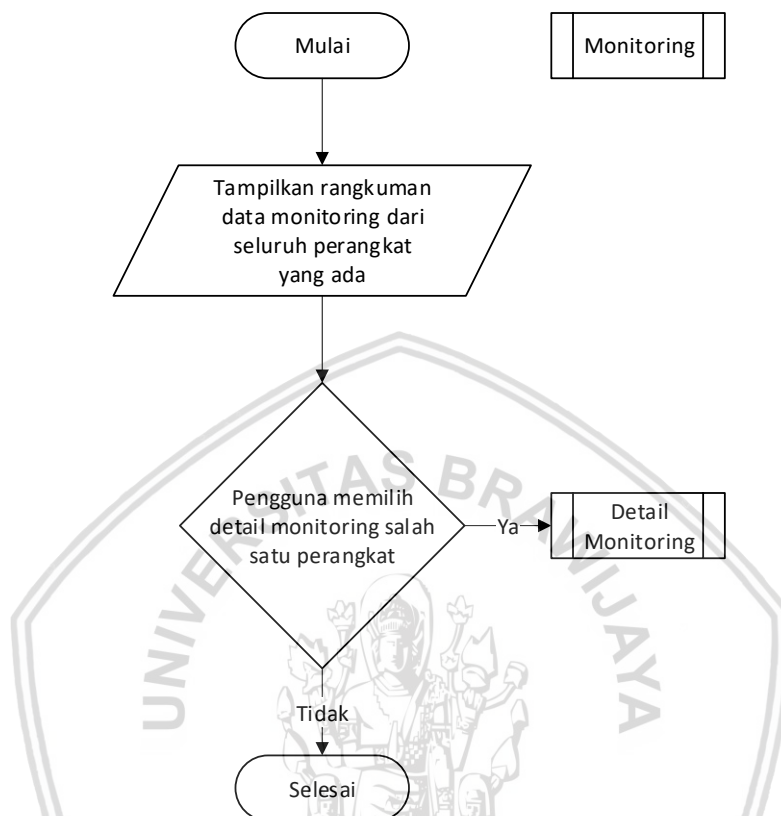
**Gambar 5.10** Flowchart Perangkat Baru dan Perangkat Terdaftar

### 3. Monitoring

Seperti pada Gambar 5.11, maka penjelasan mengenai alur kerja pada bagian *monitoring* adalah sebagai berikut:

- a. Ketika pengguna mengakses halaman *monitoring*, maka akan ditampilkan rangkuman dari *monitoring* seluruh perangkat *monitoring* yang ada. Rangkuman *monitoring* yang ditampilkan mulai dari nama perangkat *monitoring*, nama ruangan dimana perangkat tersebut berada, total penggunaan listrik dalam KWh, TDL aktif yang dipilih, dan estimasi biaya penggunaan listrik dalam rupiah.

- b. Jika pengguna ingin melihat detail *monitoring* dari salah satu perangkat *monitoring*, maka pengguna harus memilih salah satu perangkat *monitoring* yang ingin dilihat detailnya. Setelah itu, maka tampilan *web monitoring* akan berpindah ke halaman detail *monitoring* dari perangkat tersebut.

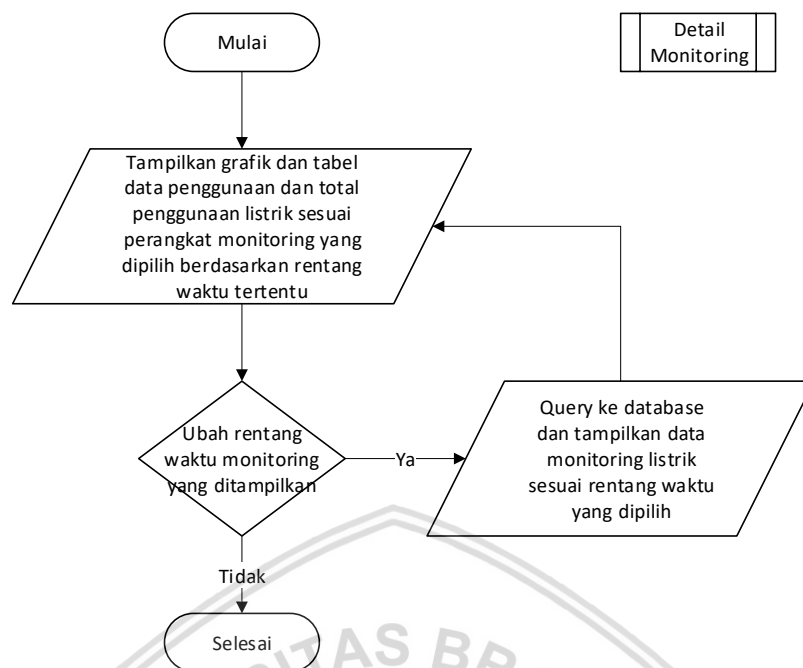


**Gambar 5.11** Flowchart Monitoring

#### 4. Detail Monitoring

Seperti pada Gambar 5.12, maka penjelasan mengenai alur kerja pada bagian detail *monitoring* adalah sebagai berikut:

- Ketika pengguna mengakses halaman detail *monitoring*, maka *web* akan melakukan *query* data *monitoring* ke tabel *monitoring* di *database server* dan menampilkan grafik dan tabel dari data penggunaan listrik (W) serta total penggunaan listrik (KWh) dari perangkat *monitoring* yang dipilih. Secara *default* rentang waktu data *monitoring* yang ditampilkan adalah data *monitoring* dalam rentang waktu satu menit terakhir.
- Apabila pengguna mengubah rentang waktu data *monitoring* yang ditampilkan, maka tampilan grafik serta tabel data penggunaan listrik dan total penggunaan listrik akan ditampilkan sesuai rentang waktu yang dipilih. Terdapat dua opsi dalam memilih rentang waktu *monitoring* yang ditampilkan, yaitu mulai dari satu menit terakhir, satu jam terakhir, hingga satu tahun terakhir, atau berdasarkan tanggal awal dan akhir yang dipilih.



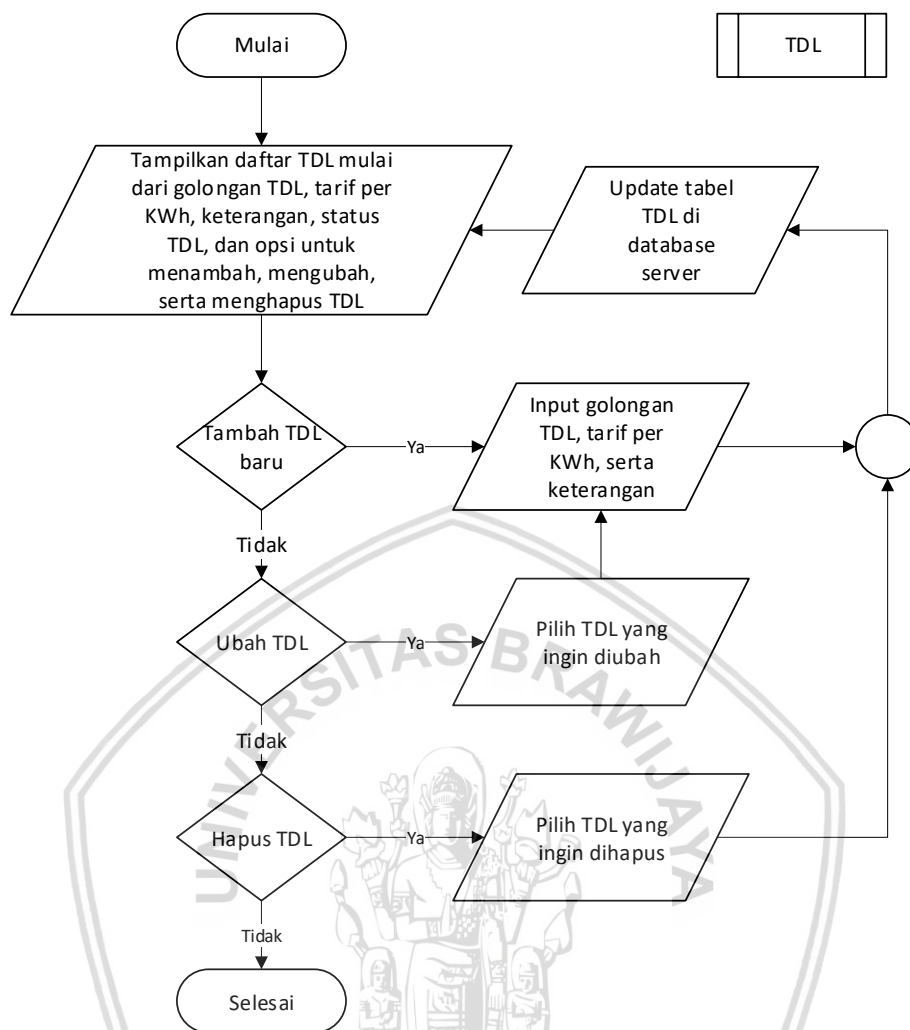
**Gambar 5.12** Flowchart Detail Monitoring

## 5. Tarif Dasar Listrik

Seperti pada Gambar 5.13, maka penjelasan mengenai alur kerja pada bagian tariff dasar listrik adalah sebagai berikut:

- Ketika pengguna mengakses halaman tarif dasar listrik, maka akan ditampilkan daftar TDL mulai dari golongan TDL, tarif listrik per KWh, keterangan apakah golongan TDL tersebut masuk pada kategori TDL yang bersubsidi atau non-subsidi, status TDL aktif yang dipilih untuk perhitungan estimasi biaya penggunaan listrik pada halaman *monitoring*, serta opsi untuk menambah, mengubah, atau menghapus TDL yang ada.
- Apabila pengguna ingin menambahkan TDL baru ke dalam daftar TDL, maka pengguna harus memasukkan golongan TDL tersebut, tarif listrik per KWh, serta memilih apakah golongan TDL tersebut bersubsidi atau non-subsidi. Setelah itu, maka TDL baru akan ditambahkan pada tabel TDL di *database server*.
- Apabila pengguna ingin mengubah TDL yang ada dalam daftar, maka pengguna harus memilih salah satu TDL yang ada dalam daftar lalu pengguna dapat mengubah golongan TDL, tarif per KWh, serta memilih golongan TDL bersubsidi atau non-subsidi. Setelah itu, maka tabel TDL pada *database server* akan berubah sesuai dengan perubahan yang dimaksud.
- Apabila pengguna ingin menghapus TDL yang ada dalam daftar, maka pengguna harus memilih TDL yang ingin dihapus. Setelah itu, TDL yang dimaksud akan dihapus pada tabel TDL di *database server*.
- Perubahan TDL hendaknya mengacu pada ketentuan yang sudah ditetapkan oleh PT. PLN Persero.





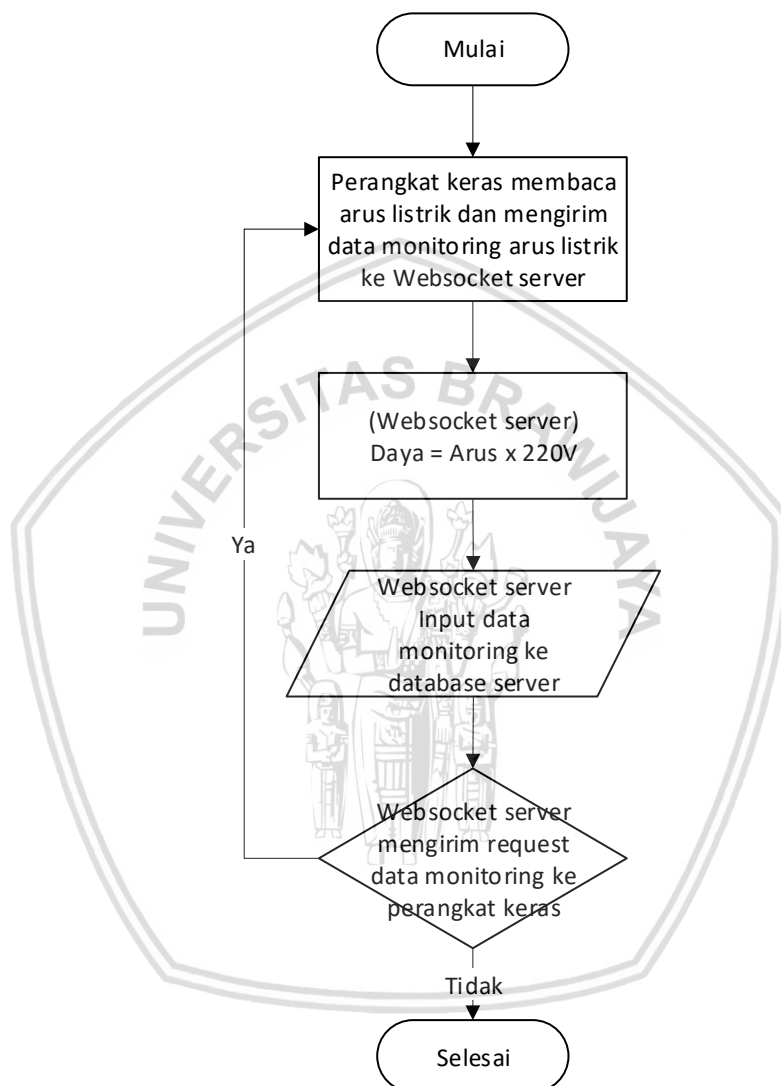
Gambar 5.13 Flowchart Tarif Dasar Listrik

### 5.1.3 Perancangan Keseluruhan Sistem

Setelah semua perancangan di sisi perangkat keras dan server selesai dilakukan, maka langkah berikutnya adalah menyatukan seluruh perancangan yang telah dibuat menjadi satu kesatuan sistem yang utuh. Adapun alur kerja keseluruhan dari sistem *monitoring* daya listrik yang dibuat berdasarkan Gambar adalah sebagai berikut:

- Pada saat perangkat *monitoring* daya listrik diaktifkan, maka perangkat keras akan mencoba terhubung dengan jaringan Wi-Fi yang sudah dikonfigurasi sebelumnya. Setelah berhasil terhubung dengan jaringan Wi-Fi, maka sensor arus listrik akan melakukan akuisisi data sinyal *analog* arus listrik yang selanjutnya akan diproses oleh mikrokontroler NodeMCU dan dikirimkan ke Websocket server.
- Websocket server menerima data arus listrik dan MAC address dari mikrokontroler NodeMCU perangkat keras lalu akan menghitung daya listrik dengan rumus  $\text{Daya} = \text{Arus} \times 220\text{V}$ .

- c. Setelah daya listrik didapatkan, maka selanjutnya Websocket server akan menyimpan data *monitoring* listrik ke *database server* sesuai dengan MAC *address* dari perangkat keras yang digunakan.
- d. Setelah data *monitoring* berhasil disimpan ke *database server*, maka Websocket server akan mengirimkan pesan *request* data *monitoring* ke perangkat keras.



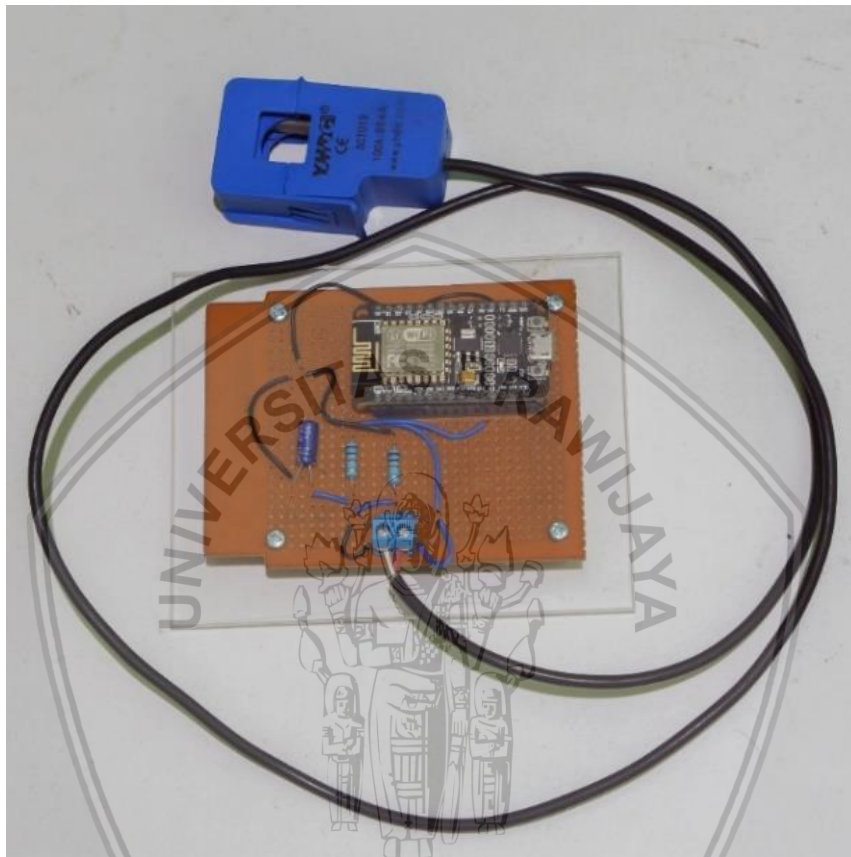
**Gambar 5.14** Alur Kerja Keseluruhan Sistem

## 5.2 Implementasi

Pada tahap ini akan dijelaskan mengenai implementasi dari perangkat keras dan perangkat lunak sistem *monitoring*. Implementasi perangkat keras meliputi perakitan perangkat keras sesuai perancangan sistem dan implementasi perangkat lunak meliputi pembuatan perangkat lunak pada perangkat *monitoring listrik*, Websocket server, database server, dan web monitoring listrik.

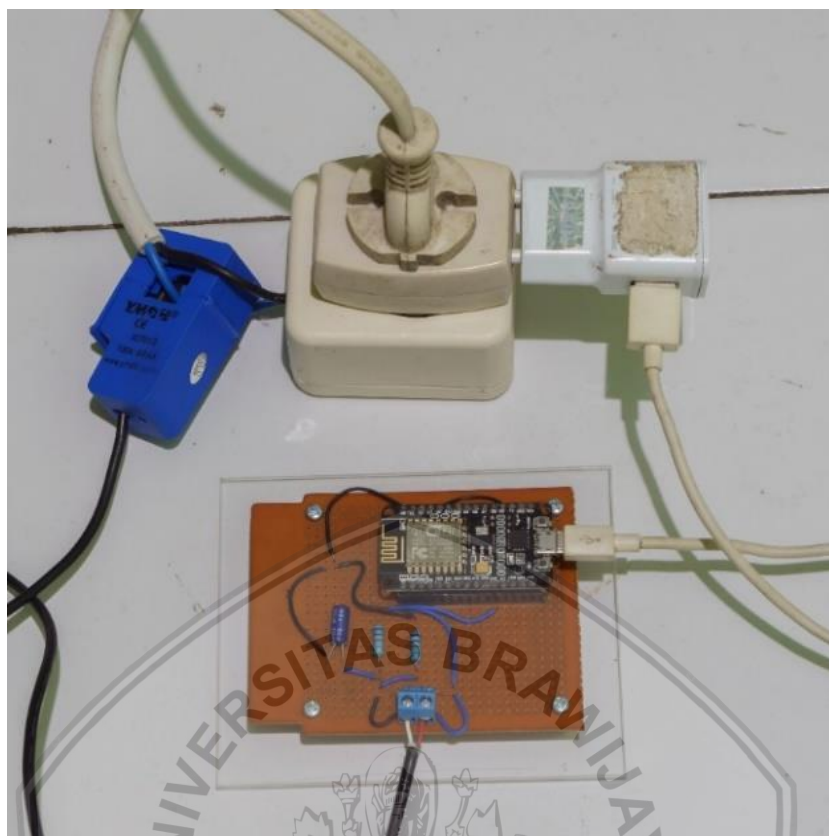
### 5.2.1 Implementasi Perangkat Keras

Setelah perangkat keras *monitoring listrik* dirancang, maka selanjutnya adalah mengimplementasikan perancangan tersebut. Tahap awal dari implementasi perangkat keras *monitoring listrik* adalah dengan menyusun rangkaian elektronik yang diperlukan sesuai rangkaian skematik pada tahap perancangan. Tampilan bentuk fisik dari perangkat keras seperti yang ditunjukkan pada Gambar 5.15.



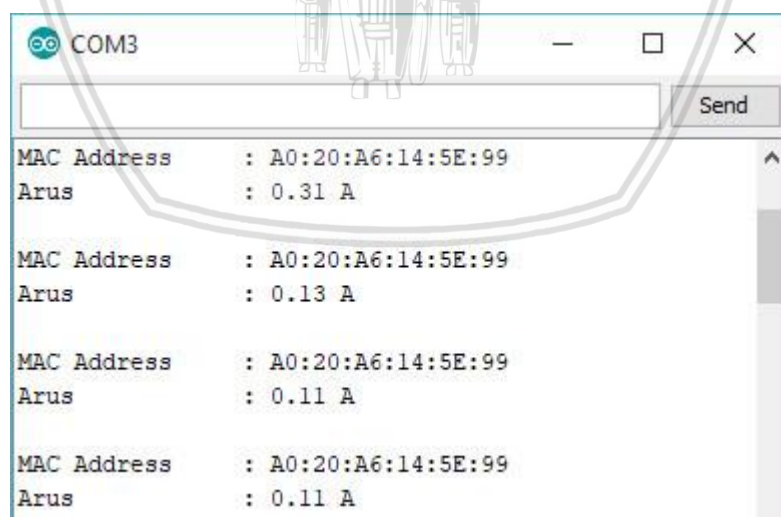
**Gambar 5.15** Bentuk fisik perangkat keras

Setelah rangkaian perangkat keras selesai dibuat, hal pertama yang dilakukan adalah menghubungkan perangkat keras dengan sumber daya listrik. Sumber daya listrik pada perangkat keras dapat diperoleh dengan cara menghubungkan *port micro USB* yang terdapat pada mikrokontroler dengan sumber daya seperti *adapter* pengisi daya ponsel 5V, *port USB* pada *laptop*, atau *power bank*. Setelah itu, untuk menguji coba perangkat keras dapat dilakukan dengan cara mengunggah (*upload*) kode program untuk perangkat keras *monitoring listrik*. Sebelumnya, pastikan rangkaian elektronik sudah sesuai rangkaian skematik perancangan dimana *pinout A0* dihubungkan dengan bagian positif dari sensor arus dan *burden resistor*, *ground* dihubungkan dengan bagian negatif, resistor pembagi tegangan, dan kapasitor yang terhubung pada sensor, VCC dihubungkan dengan *pinout 3.3V*, dan pasang sensor pada salah satu bagian kabel di terminal listrik yang ingin diukur, seperti yang ditunjukkan pada Gambar 5.16.



**Gambar 5.16** Perangkat keras terpasang pada terminal listrik

Apabila perangkat keras dan program pada perangkat keras sudah berjalan, maka pada *serial monitor* di Arduino IDE akan menampilkan hasil *monitoring* seperti yang ditunjukkan pada Gambar 5.17.



**Gambar 5.17** Output monitoring pada serial monitor Arduino IDE

Sampai pada tahap ini, maka selanjutnya adalah mengulangi proses diatas pada setiap perangkat keras *monitoring listrik* yang akan digunakan. Jika semua sudah dilakukan, maka perangkat keras *monitoring listrik* telah siap digunakan pada masing-masing ruangan yang ingin dipantau.

## 5.2.2 Implementasi Perangkat Lunak

Pada tahap ini akan dijelaskan mengenai implementasi perangkat lunak yang digunakan pada penelitian ini. Implementasi perangkat lunak lebih jelasnya akan dibagi menjadi empat bagian yaitu implementasi program pada perangkat *monitoring listrik*, *Websocket server*, *database server*, dan *web monitoring listrik*

### 5.2.2.1 Implementasi Program pada Perangkat Keras

Implementasi program pada perangkat keras *monitoring listrik* dilakukan dengan menerapkan perancangan yang telah dibuat sebelumnya. Dimulai dari inisialisasi awal program, lalu dilanjutkan dengan proses komunikasi dengan *Websocket server*, dan pembacaan arus yang dilakukan oleh sensor arus.

#### 1. Inisialisasi awal

Inisialisasi awal pada program perangkat keras terdiri dari inisialisasi *library* yang digunakan, lebar *baud rate serial*, nilai kalibrasi sensor arus listrik, serta konfigurasi dari jaringan Wi-Fi yang digunakan. Berikut merupakan potongan kode sumber dari program pada kategori inisialisasi awal:

**Tabel 5.5** Potongan kode sumber kategori inisialisasi awal

1	#include <ArduinoJson.h>
2	#include <StreamString.h>
3	#include <ESP8266WiFi.h>
4	#include <ESP8266WiFiMulti.h>
5	#include <WebSocketsClient.h>
6	#include "EmonLib.h"
7	String mac = WiFi.macAddress();
8	EnergyMonitor emon1;
9	ESP8266WiFiMulti WiFiMulti;
10	WebSocketsClient webSocket;
	...
11	void setup() {
12	Serial.begin(9600);
13	emon1.current(A0, 90.9);
14	WiFiMulti.addAP("PD.Muncul.Jaya", "s5up3ermii1");
15	while (WiFiMulti.run() != WL_CONNECTED) {
16	delay(100);
17	}
18	webSocket.begin("192.168.1.64", 5000);
19	webSocket.onEvent(webSocketEvent);
20	}
21	void loop() {



22	<code>websocket.loop();</code>
23	<code>}</code>

Berikut merupakan penjelasan dari potongan kode sumber pada Tabel 5.5:

- a. Baris ke-1 hingga baris ke-6 merupakan inisialisasi *library* yang digunakan pada program ini. Pada program ini menggunakan *library* ArduinoJson karena data *monitoring* akan dikemas dalam bentuk format json agar *server* dapat membedakan setiap variabel data yang dikirim. Selain itu *library* StreamString digunakan untuk mengirim objek-objek json menjadi satu variabel, karena *library* Websocket yang terdapat di Arduino hanya mendukung pengiriman data dengan format *string*. Untuk konfigurasi Wi-Fi sebagai media penghubung antara perangkat keras dengan *server* pada program ini membutuhkan *library* ESP8266WiFi dan ESP8266WiFiMulti. *Library* WebSocketsClient digunakan karena pada pengiriman data dari perangkat keras ke *server* menggunakan protokol komunikasi Websocket. Sedangkan *library* EmonLib digunakan untuk proses perhitungan nilai arus listrik yang dibaca oleh sensor arus, karena sensor arus listrik membaca arus listrik dalam bentuk sinyal *analog*. Oleh karena itu, sinyal *analog* tersebut perlu diubah ke dalam bentuk akhir nilai arus listrik.
- b. Baris ke-7 merupakan inisialisasi MAC *address* dari mikrokontroler NodeMCU disimpan pada variabel *mac* dengan tipe data *string*. Baris ke-8 adalah inisialisasi fungsi EnergyMonitor pada *library* EmonLib yang selanjutnya akan disebut *emon1*. Baris ke-9 merupakan fungsi ESP8266WiFiMulti pada *library* ESP8266WiFiMulti yang selanjutnya akan disebut *WiFiMulti*. Sedangkan WebSocketsClient pada fungsi WebSocketsClient selanjutnya akan disebut *websocket* di program ini.
- c. Baris ke-11 hingga baris ke-20 merupakan fungsi *setup* dari program.
- d. Baris ke-12 inisialisasi *serial* dilakukan dengan *baud rate* sebesar 9600 bps.
- e. Baris ke-13 merupakan inisialisasi *pin analog* dari mikrokontroler NodeMCU yang digunakan oleh sensor arus listrik dan 90.9 merupakan nilai kalibrasi sesuai perhitungan yang telah dilakukan pada tahap perancangan.
- f. Baris ke-14 hingga baris ke-17 merupakan konfigurasi dan proses koneksi ke jaringan Wi-Fi yang digunakan. PD.Muncul.Jaya merupakan SSID dari jaringan Wi-Fi yang digunakan, sedangkan *s5up3ermii1* merupakan *password* dari jaringan Wi-Fi yang digunakan. Fungsi *WiFiMulti.run()* digunakan untuk mengecek apakah koneksi Wi-Fi antara mikrokontroler NodeMCU dengan *wireless router* telah terjalin. Jika belum, maka akan mencoba kembali proses koneksi dengan *wireless router* dengan jeda waktu 100 milidetik hingga status dari fungsi *WiFiMulti.run()* memiliki *output* status *WL\_CONNECTED* yang berarti koneksi Wi-Fi telah berhasil terjalin.
- g. Baris ke-18 hingga baris ke-19 merupakan fungsi Websocket dari program dimana 192.168.1.64 dan 5000 merupakan alamat IP dan *port* dari Websocket *server* yang digunakan. Nantinya mikrokontroler NodeMCU akan mengirimkan

data *monitoring* arus listrik yang dilakukan oleh sensor arus listrik ke alamat tersebut.

- h. Baris ke-21 hingga baris ke-23 merupakan fungsi *loop* dimana pada fungsi *loop* tersebut terdapat fungsi `websocket.loop()` yang berarti Websocket akan berjalan terus menerus tanpa henti.

## 2. Websocket

Fungsi Websocket pada program ini merupakan proses komunikasi yang terjadi antara mikrokontroler NodeMCU dengan Websocket *server*. Pada fungsi Websocket, komunikasi antara mikrokontroler NodeMCU dengan Websocket *server* dibagi menjadi beberapa kondisi diantaranya ketika koneksi terputus, terhubung, dan terjadi pertukaran data antara keduanya.

**Tabel 5.6** Potongan kode sumber kategori Websocket

```

1 void websocketEvent(WStype_t type, uint8_t * payload, size_t length){
2     switch (type) {
3         case WStype_DISCONNECTED:
4             Serial.printf("Disconnected!\n");
5             break;
6         case WStype_CONNECTED: {
7             bacaarus();
8             break;
9         }
10        case WStype_TEXT:
11            int a = 1;
12            String payload_str = String((char *) payload);
13            do {
14                if (payload_str == "minta") {
15                    bacaarus();
16                    break;
17                }
18            } while (a != 1);
19        }
20    }

```

Berikut adalah penjelasan dari kode sumber pada Tabel 5.6:

- Baris ke-3 hingga ke-5 merupakan kondisi ketika koneksi Websocket antara mikrokontroler NodeMCU dengan Websocket *server* terputus. Ketika komunikasi terputus, maka pada *serial monitor* Arduino IDE akan mencetak pesan "Disconnected".
- Baris ke-6 hingga ke-9 merupakan kondisi ketika mikrokontroler NodeMCU berhasil menjalin koneksi Websocket dengan Websocket *server* untuk

pertama kalinya. Ketika itu, mikrokontroler NodeMCU akan menjalankan fungsi baca arus yang selanjutnya akan dijelaskan pada kategori baca arus.

- c. Baris ke-10 hingga baris ke-19 merupakan kondisi ketika mikrokontroler NodeMCU menerima pesan yang dikirimkan oleh Websocket server. Apabila pesan yang diterima adalah "minta", maka mikrokontroler NodeMCU akan menjalankan fungsi baca arus. Terdapat *do-while loop* agar proses fungsi baca arus dapat dilakukan secara terus menerus tanpa perlu menjalankan ulang program dari awal.

### 3. Baca arus

Baca arus merupakan fungsi pada program yang bertindak untuk mendapatkan nilai arus listrik dari pembacaan sinyal *analog* arus listrik oleh sensor. Pada fungsi baca arus juga terjadi mekanisme pengiriman data *monitoring* arus listrik ke Websocket server.

**Tabel 5.7** Potongan kode sumber kategori baca arus

1	void bacaarus() {
2	DynamicJsonBuffer jsonBuffer(100);
3	JsonObject& data = jsonBuffer.createObject();
4	double Irms = emon1.calcIrms(1480);
5	data["id"] = mac;
6	data["value"] = Irms;
7	StreamString databuf;
8	data.printTo(databuf);
9	websocket.sendTXT(databuf);
10	Serial.print("MAC Address\t: ");
11	Serial.println(mac);
12	Serial.print("Arus\t\t: ");
13	Serial.print(Irms);
14	Serial.println(" A");
15	delay(1000);
16	}

Berikut merupakan penjelasan dari kode sumber pada Tabel 5.7:

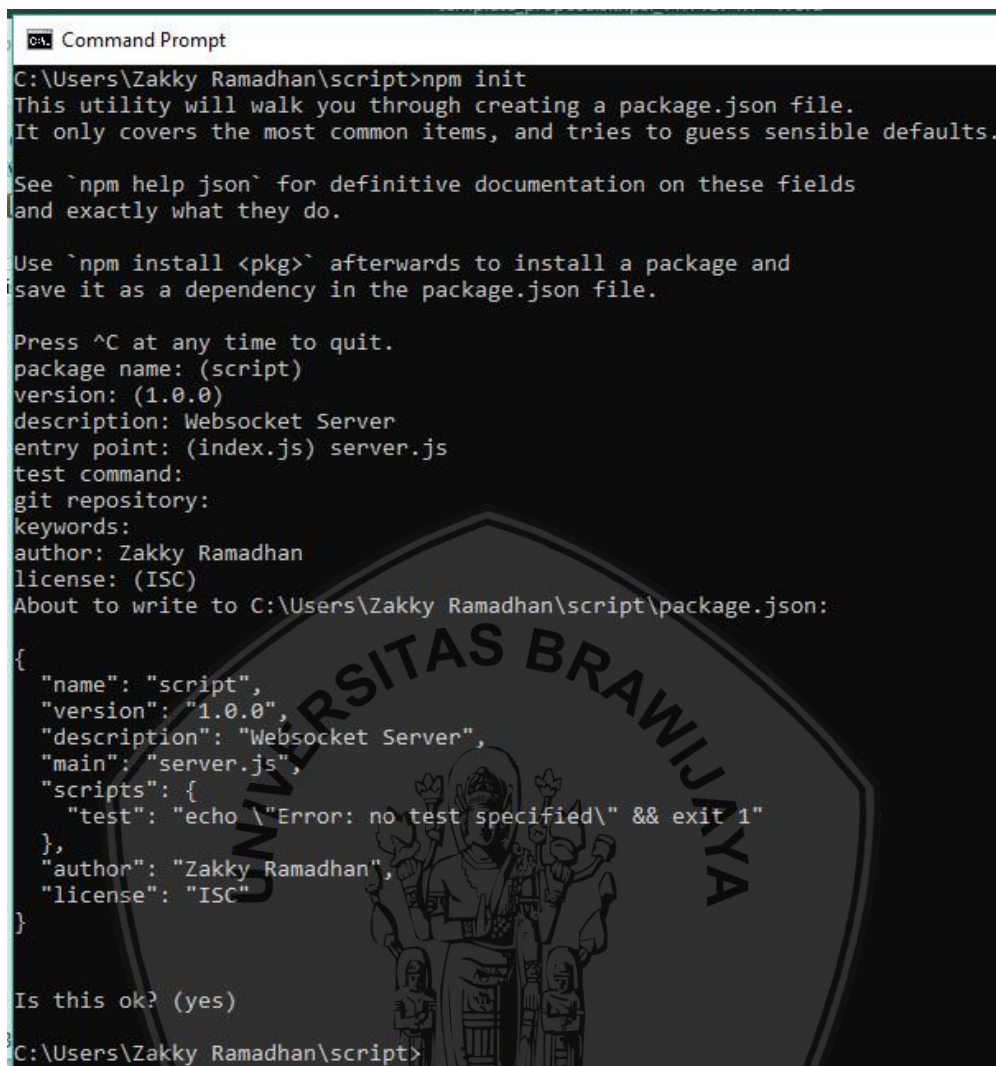
- a. Baris ke-2 merupakan inisialisasi memori untuk *buffer* json sebesar 100 bytes.
- b. Baris ke-3 merupakan pembuatan objek json dengan nama objek "data".
- c. Baris ke-4 merupakan proses *sampling* sinyal *analog* arus listrik yang diperoleh sensor arus listrik dengan memanfaatkan fungsi `emon1.calcIrms`. Sebanyak 1480 sampel sinyal *analog* dari arus listrik digunakan oleh fungsi `emon1.calcIrms` untuk mendapatkan nilai arus listrik, yang mana 1480 sampel tersebut merupakan banyaknya sampel yang dibutuhkan oleh fungsi

- emon1.calcIrms secara *default*. Data nilai arus listrik disimpan pada variabel *Irms*.
- Baris ke-5 dan ke-6 merupakan pembuatan objek json *MAC address* dari mikrokontroler NodeMCU yang digunakan dengan nama objek json “id” dan objek json nilai arus listrik yang sudah dihitung dengan menggunakan fungsi *emon1.calcIrms* sebelumnya dengan nama objek json “Irms”.
  - Baris ke-7 hingga baris ke-9 merupakan proses pengemasan objek-objek json yang telah dibuat sebelumnya ke dalam bentuk *StreamString* dengan nama variabel “*databuf*”. Selanjutnya variabel “*databuf*” tersebut dikirim ke *Websocket server*. Pengiriman dilakukan dalam bentuk format data *StreamString* karena *library* *Websocket* di *Arduino* yang digunakan pada penelitian ini hanya mendukung pengiriman data dalam bentuk *String*.
  - Baris ke-10 hingga baris ke-14 pada *serial monitor* *Arduino IDE* akan mencetak *MAC address* dari mikrokontroler NodeMCU yang digunakan dan nilai arus listrik yang didapat.
  - Pada program ini menggunakan *delay* selama 1000 milidetik agar data *monitoring* arus listrik yang dikirim ke *Websocket server* dikirim dalam *interval* waktu selama satu detik atau dengan kata lain data *monitoring* arus listrik setiap detiknya akan dikirim ke *Websocket server*.

#### 5.2.2.2 Implementasi Websocket Server

*Websocket server* bertugas untuk mengirimkan *request* data arus listrik ke perangkat *monitoring* listrik serta menerima data yang dikirimkan oleh perangkat *monitoring* listrik untuk selanjutnya data tersebut akan dikirim dan disimpan pada *database server*. Pada penelitian ini, implementasi *Websocket server* diterapkan pada *server* dengan menggunakan perangkat lunak *node.js* yang dipasang pada *server* dengan sistem operasi *Microsoft Windows 10 64-bit*. Untuk dapat menjalankan *Websocket server*, sebelumnya kita harus melakukan inisialisasi awal terhadap *Websocket server* yang nantinya akan digunakan. Hal tersebut dapat dilakukan dengan menjalankan perintah **npm init** di *command prompt* dengan sebelumnya membuat *folder* yang akan dijadikan lokasi dari *Websocket server*. Pada penelitian ini, *Websocket server* berada pada sebuah *folder* dengan nama “*script*”.

Berdasarkan pada Gambar 5.18, inisialisasi awal yang dilakukan diantaranya menentukan *package name* dengan nama “*script*”, *entry point* yang merupakan *file* *JavaScript* dari *Websocket server* yang digunakan dengan nama “*server.js*”, dan *author* dengan nama “*Zakky Ramadhan*”. Jika inisialisasi awal berhasil dilakukan, maka pada *folder* “*script*” tersebut akan muncul *file* json dengan nama “*package.json*”.



```

C:\Users\Zakky Ramadhan\script>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

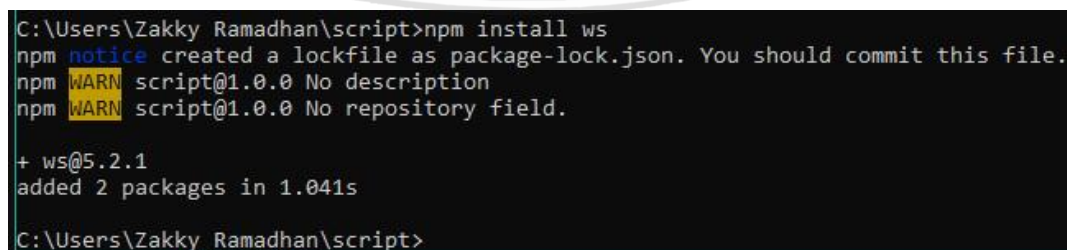
Press ^C at any time to quit.
package name: (script)
version: (1.0.0)
description: Websocket Server
entry point: (index.js) server.js
test command:
git repository:
keywords:
author: Zakky Ramadhan
license: (ISC)
About to write to C:\Users\Zakky Ramadhan\script\package.json:
{
  "name": "script",
  "version": "1.0.0",
  "description": "Websocket Server",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Zakky Ramadhan",
  "license": "ISC"
}

Is this ok? (yes)
C:\Users\Zakky Ramadhan\script>

```

**Gambar 5.18** Proses inisialisasi awal node.js Websocket server

Langkah selanjutnya setelah inisialisasi awal adalah dengan memasang modul Websocket agar Websocket server dapat berjalan. Hal tersebut dapat dilakukan dengan menjalankan perintah **npm install ws** seperti pada Gambar 5.19.



```

C:\Users\Zakky Ramadhan\script>npm install ws
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN script@1.0.0 No description
npm WARN script@1.0.0 No repository field.

+ ws@5.2.1
added 2 packages in 1.041s
C:\Users\Zakky Ramadhan\script>

```

**Gambar 5.19** Pemasangan modul Websocket

Setelah modul Websocket terpasang, langkah selanjutnya adalah memasang modul bernama nodemon. Modul nodemon ini memiliki fungsi yang sama dengan modul node (*default* node.js), dengan tambahan fitur seperti *automatic reloading* yang mana setiap perubahan yang dilakukan pada server node.js akan dimuat ulang program JavaScript server secara otomatis tanpa perlu melakukan proses



tersebut secara manual. Pada modul nodemon juga terdapat fitur *verbose log* dimana setiap proses yang berjalan pada node.js akan ditampilkan dalam bentuk *log* di *command prompt*. Untuk memasang modul nodemon dapat dilakukan dengan menjalankan perintah **npm install -g nodemon** seperti pada Gambar 5.20.

```
C:\Users\Zakky Ramadhan\script>npm install -g nodemon
C:\Users\Zakky Ramadhan\AppData\Roaming\npm\nodemon -> C:\Users\Zakky Ramadhan\AppData\Roaming\npm\node_modules\nodemon\
bin\nodemon.js

> nodemon@1.17.5 postinstall C:\Users\Zakky Ramadhan\AppData\Roaming\npm\node_modules\nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
> https://opencollective.com/nodemon/donate

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\nodemon\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ nodemon@1.17.5
added 167 packages, removed 28 packages, updated 39 packages and moved 2 packages in 15.647s
C:\Users\Zakky Ramadhan\script>
```

**Gambar 5.20** Pemasangan modul nodemon

Setelah langkah-langkah diatas selesai dilakukan, maka langkah berikutnya adalah membuat kode sumber program Websocket server dengan bahasa JavaScript. Program Websocket server pada penelitian ini disimpan dengan nama **server.js** yang mana program tersebut akan dibagi menjadi beberapa bagian, diantaranya:

## 1. Inisialisasi Server dan Penanggalan

Proses inisialisasi awal yang terjadi pada Websocket meliputi inisialisasi alamat dan *port* dari Websocket server. Pada penelitian ini Websocket server berjalan pada alamat IP localhost dan *port* 5000. Sedangkan untuk *database server* tujuan, berada pada alamat IP yang sama pula dengan Websocket server.

**Tabel 5.8** Potongan kode sumber inisialisasi server dan penanggalan

1	var server = require('ws').Server;
2	var sock = new server({port:5000});
3	var mysql = require('mysql');
4	var con = mysql.createConnection({
5	host: "localhost",
6	user: "root",
7	password: "",
8	database: "monitoring_listrik"
9	});
10	Date.prototype.bulan = function(){
11	var retval = (this.getMonth() + 1);
12	if (retval < 10){
13	return ("0" + retval.toString());
14	}
15	else{

```
16         return retval.toString();
17     }
18 }
19 Date.prototype.bulansblm = function(){
20     var retval = (this.getMonth());
21     if (retval < 10){
22         return ("0" + retval.toString());
23     }
24     else{
25         return retval.toString();
26     }
27 }
28 Date.prototype.bulansdh = function(){
29     var retval = (this.getMonth() + 2);
30     if (retval < 10){
31         return ("0" + retval.toString());
32     }
33     else{
34         return retval.toString();
35     }
36 }
37 Date.prototype.tanggal = function(){
38     var retval = this.getDate();
39     if (retval < 10){
40         return ("0" + retval.toString());
41     }
42     else{
43         return retval.toString();
44     }
45 }
46 Date.prototype.jam = function(){
47     var retval = this.getHours();
48     if (retval < 10){
49         return ("0" + retval.toString());
50     }
51     else{
52         return retval.toString();
53     }
54 }
55 Date.prototype.menit = function(){
56     var retval = this.getMinutes();
```

```

57     if (retval < 10){
58         return ("0" + retval.toString());
59     }
60     else{
61         return retval.toString();
62     }
63 }
64 Date.prototype.detik = function(){
65     var retval = this.getSeconds();
66     if (retval < 10){
67         return ("0" + retval.toString());
68     }
69     else{
70         return retval.toString();
71     }
72 }
73 Date.prototype.milidetik = function(){
74     var retval = this.getMilliseconds();
75     if (retval < 10){
76         return ("0" + retval.toString());
77     }
78     else{
79         return retval.toString();
80     }
81 }

```

Berikut merupakan penjelasan dari kode sumber pada Tabel 5.8:

- Baris ke-1 hingga baris ke-9 merupakan inisialisasi dari Websocket server dan database server tujuan. Websocket server berjalan pada localhost dengan alamat IP 192.168.1.64 dan port 5000. Untuk database server juga berjalan pada alamat IP yang sama dengan Websocket server dengan konfigurasi server MySQL, username "root", tanpa password, dan nama database "monitoring\_listrik".
- Baris ke-10 hingga baris ke-81 merupakan inisialisasi penanggalan server yang terdiri dari tanggal, bulan, tahun, jam, menit, detik, dan milidetik. Untuk bagian bulan dibagi menjadi tiga bagian, yaitu bulan saat ini, bulan sebelum, dan bulan sesudah. Hal tersebut dilakukan karena nantinya pada tampilan estimasi biaya penggunaan listrik hanya menampilkan data estimasi biaya penggunaan listrik pada bulan saat itu saja.

## 2. Komunikasi Websocket pada Websocket Server

Websocket server memiliki keterkaitan dengan database server dan perangkat keras monitoring listrik, dimana perangkat keras akan mengirimkan data

*monitoring* listrik ke Websocket server yang selanjutnya data *monitoring* tersebut akan diteruskan ke *database server* untuk disimpan. Websocket server juga bertugas untuk mengirimkan *request* data *monitoring* ke perangkat keras.

**Tabel 5.9** Potongan kode sumber Websocket pada Websocket server

1	sock.on('connection', function (ws) {
2	ws.on('message', function (message) {
3	var d = new Date();
4	var dt = d.getFullYear()+'-'+d.bulan()+'-'+d.tanggal()+''
5	'+d.jam()+':'+d.menit()+':'+d.detik();
6	var data = JSON.parse(message);
7	var daya = data.value * 220;
8	con.connect(function(err) {
9	con.query("SELECT * FROM list_device WHERE mac =
10	'"+data.id+"', function (err, result, fields) {
11	if (err) throw err;
12	if (result == "") {
13	con.query("SELECT * FROM temp_mac WHERE mac =
14	'"+data.id+"', function (err, hasil){
15	if (err) throw err;
16	if (hasil == "") {
17	con.query("INSERT INTO temp_mac (mac) VALUES
18	('"+data.id+"', function (err, hasil1){
19	console.log(data.id+' berhasil di simpan ke temp');
20	});
21	}
22	});
23	}
24	else{
25	con.query("SELECT id_device AS device, id_ruangan AS
26	room FROM list_device WHERE mac =
27	'"+data.id+"', function (err, hasil2){
28	if (err) throw err;
29	ws.send('minta');
30	var sql1 = 'SELECT AVG(daya) AS dy FROM monitoring WHERE
31	reset = "1";
32	con.query(sql1, function (err, result3) {
33	if (err) throw err;
34	var sql2 = 'SELECT COUNT(daya) AS sec FROM monitoring
35	WHERE reset = "1";

```

36     con.query(sql2, function (err, result4) {
37         if (err) throw err;
38         var kwh = (result3[0].dy/1000)*(result4[0].sec/3600);
39         console.log("MAC   : " + data.id);
40         console.log("Arus  : " + data.value + " A");
41         console.log("Daya  : " + daya + " W");
42         console.log("KWh   : " + kwh + " KWh");
43         if (d.bulan() != d.bulansblm() && d.bulan() !=
44             d.bulansdh()) {
45             var sql = 'INSERT INTO monitoring (id_device,
46                 id_ruangan, arus, daya,kwh, tgl_log, reset)
47                 VALUES ("'+hasil2[0].device+'",
48                     "'+hasil2[0].room+'", "'+data.value+'", "'
49                     +daya+'", "'+kwh+'", "'+dt+'", "1")';
50             var d1 = new Date();
51             var dt1 = d1.getFullYear()+'-'+d1.bulan()+'-'
52                 +d1.tanggal()+' '+d1.jam()+':'+d1.menit()+':
53                 +d1.detik()+':'+d1.milidetik();
54             console.log(dt1);
55             console.log("");
56         }else{
57             var sql = 'INSERT INTO monitoring (id_device,
58                 id_ruangan, arus, daya,kwh, tgl_log, reset)
59                 VALUES ("'+hasil2[0].device+'",
60                     "'+hasil2[0].room+'",
61                     "'+data.value+'", "'+daya+'", "'+kwh+'",
62                     "'+dt+'", "0")';
63             var d1 = new Date();
64             var dt1 = d1.getFullYear()+'-'+d1.bulan()+'-'
65                 +d1.tanggal()+' '+d1.jam()+':'+d1.menit()+':
66                 +d1.detik()+':'+d1.milidetik();
67             console.log(dt1);
68             console.log("");
69             }
70         if (d.bulan() > d.bulansblm()) {
71             var sql_update = "UPDATE monitoring SET reset = '0'
72                 WHERE MONTH(tgl_log) < MONTH(CURRENT_DATE())";
73             con.query(sql_update, function (err, result5) {
74                 if (err) throw err;
75                 });
76             con.query(sql, function (err, result2) {

```



```

77         if (err) throw err;
78     });
79     }
80     });
81     });
82     });
83     }
84     });
85     });
86     });
87     ws.on('close', function(){
88         console.log('Client disconnected');
89     });
90     console.log('Client connected');
91 });

```

Berikut penjelasan dari kode sumber pada Tabel 5.9:

- Baris pertama merupakan proses pembuatan *socket* baru ketika ada perangkat *monitoring* listrik yang terhubung dengan Websocket *server*.
- Pada baris ke-2 hingga baris ke-7, Websocket *server* akan melakukan inisialisasi waktu, *parsing* (pembongkaran) pesan json yang diterima dari perangkat *monitoring* listrik, serta menghitung nilai daya dengan mengalikan nilai arus listrik yang diterima dengan 220 V. Inisialisasi waktu dilakukan karena nantinya data yang tadi diterima dan dihitung akan dimasukkan ke *database server*, dimana waktu tersebut selanjutnya berguna untuk proses perhitungan KWh. *Parsing* pesan disimpan dalam variabel dengan nama “data” dan nilai daya listrik dalam variabel “daya”.
- Pada baris ke-8, Websocket *server* akan mencoba terhubung dengan *database server*.
- Baris ke-9 hingga baris ke-23 adalah proses pengecekan informasi perangkat *monitoring* listrik apakah sudah terdaftar di dalam perangkat terdaftar atau belum. Jika perangkat belum ada di dalam daftar perangkat terdaftar, maka perangkat tersebut teridentifikasi sebagai perangkat baru yang selanjutnya MAC *address* dari perangkat tersebut akan disimpan pada tabel “temp\_mac” di *database server*.
- Jika perangkat *monitoring* listrik sudah terdaftar di *database server*, maka program selanjutnya akan berlanjut pada baris ke-24 hingga baris ke-74. Pada bagian tersebut, terdapat mekanisme *input* data *monitoring* listrik ke *database server* dan proses *request* data *monitoring* listrik ke perangkat keras.
- Pada baris ke-25 hingga baris ke-29, Websocket *server* akan melakukan *query* informasi MAC *address* ke *database server* sesuai dengan pesan MAC *address* yang diterima dari perangkat *monitoring* listrik. Setelah itu, Websocket *server*

- akan mengirimkan *request* data *monitoring* listrik ke perangkat keras yang sesuai dengan *MAC address* tersebut.
- g. Pada baris ke-30 hingga baris ke-31, *Websocket server* akan melakukan perhitungan rerata nilai daya listrik pada bulan terakhir yang terdapat di *database server* sesuai dengan *MAC address* yang dimaksud. Data bulan terakhir yang dimaksud dilihat pada variabel *reset* yang berada di *database server*. Bulan terakhir ditandai dengan nilai variabel *reset* bernilai 1.
  - h. Setelah menghitung rerata nilai daya listrik, maka pada baris ke-34 hingga baris ke-35 *Websocket server* akan melakukan perhitungan lamanya penggunaan listrik dari sebuah perangkat *monitoring* listrik pada bulan terakhir. Lamanya penggunaan listrik didapat dari menghitung jumlah data *monitoring* tiap detik yang ada pada *database server* dari *MAC address* yang dimaksud dan bulan terakhir juga ditentukan dari variabel *reset* di *database server* yang bernilai 1.
  - i. Pada baris ke-38, *Websocket server* akan melakukan perhitungan nilai KWh dari sebuah perangkat *monitoring* listrik yang dimaksud dengan membagi rerata nilai daya listrik dengan 1.000, lalu mengalikannya dengan lamanya penggunaan listrik yang sudah dibagi dengan 3.600. Rerata nilai daya listrik dibagi dengan 1.000 karena pada nilai daya listrik yang terdapat pada *database server* disimpan dalam satuan Watt, maka perlu dibagi 1.000 agar satuan menjadi Kilowatt. Sedangkan lamanya penggunaan listrik perlu dibagi dengan 3.600 karena lamanya penggunaan listrik menggunakan satuan sekon, maka dari itu perlu dibagi dengan 3.600 agar satuan menjadi jam (*hour*).
  - j. Baris ke-39 hingga baris ke-42 pada *console log node.js Websocket server* akan mencetak *MAC address* dari perangkat *monitoring* listrik, nilai arus listrik yang didapat dalam satuan ampere, daya listrik dalam satuan Watt, serta total penggunaan daya listrik dalam satuan KWh.
  - k. Pada baris ke-43 hingga baris ke-83, *Websocket server* akan melakukan *input* data *monitoring* ke *database server* yang terdiri dari *ID device*, *ID ruangan*, nilai arus listrik yang dibaca sensor, perhitungan daya dan KWh, waktu data di-*input* ke *database server*, serta variabel *reset* bernilai 1. Apabila pada suatu waktu sudah berganti ke bulan berikutnya, maka pada baris ke-71 hingga baris ke-75 *Websocket server* akan melakukan *update* seluruh variabel *reset* pada *database server* menjadi 0 untuk bulan-bulan sebelum bulan saat ini.
  - l. Pada *log Websocket server* akan menampilkan pesan "*Client Disconnected*" ketika koneksi *Websocket* dengan perangkat *monitoring* listrik terputus. Begitu pula ketika perangkat *monitoring* listrik terhubung dengan *Websocket server*, maka pada *log Websocket server* akan mencetak "*Client Connected*" seperti yang ditunjukkan pada baris ke-87 hingga baris ke-91.

Untuk menjalankan *node.js Websocket server*, dapat dilakukan dengan cara menjalankan perintah **nodemon server.js** atau **node server.js** di *command prompt*. Berdasarkan Gambar 5.21, ketika perangkat *monitoring* listrik berhasil terhubung dengan *Websocket server*, maka akan muncul *log MAC address* dari

mikrokontroler NodeMCU dari perangkat keras, nilai arus listrik, dan waktu data *monitoring* dimasukkan ke *database server*.

```

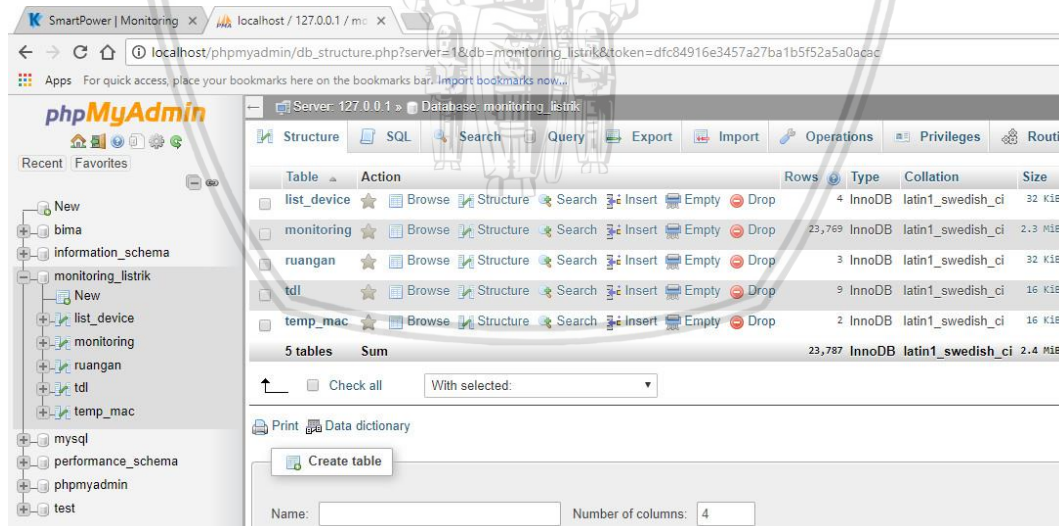
C:\Users\Zakky Ramadhan\script>nodemon server.js
[nodemon] 1.17.5
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`

[Websocket] server started
Client connected
MAC : A0:20:A6:14:5E:99
Arus : 0.11 A
Daya : 24.2 W
KWh : 0.2569361843200814 KWh
2018-07-16 20:56:42:523
  
```

Gambar 5.21 Websocket server node.js yang sedang berjalan

### 5.2.2.3 Implementasi Database Server

Implementasi *database server* diterapkan pada *server* dengan menggunakan perangkat lunak phpmyadmin dan MySQL *database server* yang sudah dipasang sebelumnya pada *server* dengan sistem operasi Microsoft Windows 10 64-bit. Untuk mengakses *database server* dapat dilakukan dengan mengakses halaman <http://localhost/phpmyadmin> di *browser* yang terdapat di *server*.



Gambar 5.22 Database Server Sistem Monitoring Listrik

Seperti yang sudah dijelaskan pada perancangan *database server* di bagian perancangan perangkat lunak sebelumnya, di dalam *database* terdapat beberapa tabel diantaranya tabel *list\_device*, *monitoring*, *ruangan*, *tdl*, dan *temp\_mac* seperti pada Gambar 5.22. *Database monitoring* listrik pada penelitian ini disimpan dengan nama *database monitoring\_listrik*.

*Database server* dalam penelitian ini diperlukan agar data *monitoring* listrik yang diperoleh dari perangkat *monitoring* listrik dapat disimpan dan selanjutnya dapat ditampilkan pada *web monitoring* daya listrik.

## 1. Tabel *list\_device*

Showing rows 0 - 3 (4 total, Query took 0.0004 seconds.)

SELECT \* FROM `list\_device`

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

	id_device	mac	nama	id_ruangan	tgl_daftar
<input type="checkbox"/> Edit Copy Delete	10	A0:20:A6:14:5E:99	Heky Depan	9	2017-12-16 20:27:48
<input type="checkbox"/> Edit Copy Delete	11	5C:CF:7F:D5:20:4F	Ical Depan	10	2017-12-16 20:50:38
<input type="checkbox"/> Edit Copy Delete	12	A0:20:A6:05:E6:92	Jack Depan	11	2017-12-17 13:13:00
<input type="checkbox"/> Edit Copy Delete	13	5C:CF:7F:D5:1F:D8	Jack Belakang	11	2017-12-19 11:09:11

**Gambar 5.23** Tabel *list\_device* di *database server*

Sesuai perancangan yang telah dilakukan, maka seperti pada Gambar 5.23 tabel *list\_device* terdiri dari ID *device*, MAC *address* dari mikrokontroler NodeMCU perangkat *monitoring* listrik yang digunakan, nama perangkat *monitoring* listrik, ID ruangan, dan tanggal waktu perangkat tersebut didaftarkan ke *database server*.

## 2. Tabel *monitoring*

Showing rows 23750 - 23768 (23769 total, Query took 0.0158 seconds.) [id\_monitoring: 5894... - 5876...]

SELECT \* FROM `monitoring` ORDER BY `monitoring`.`id\_monitoring` DESC

<< < 951 | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

	id_monitoring	id_device	id_ruangan	arus	tegangan	daya	kwh	tgl_log	reset
<input type="checkbox"/> Edit Copy Delete	5894	11	10	0.59	220	129.8	0.00893139	2017-11-17 11:10:39	0
<input type="checkbox"/> Edit Copy Delete	5893	11	10	0.56	220	123.2	0.00889717	2017-11-17 11:10:38	0
<input type="checkbox"/> Edit Copy Delete	5892	11	10	0.65	220	143	0.00885744	2017-11-17 11:10:37	0
<input type="checkbox"/> Edit Copy Delete	5891	11	10	0.55	220	121	0.00882383	2017-11-17 11:10:35	0

**Gambar 5.24** Tabel *monitoring* di *database server*

Pada tabel *monitoring* yang telah dibuat sesuai dengan perancangan, seperti yang ditunjukkan pada Gambar 5.24 terdapat ID *monitoring* dari setiap data *monitoring* listrik yang di-input ke *database server*, ID *device* dari perangkat *monitoring* listrik yang digunakan, ID ruangan dimana perangkat tersebut ditempatkan, nilai arus listrik yang dibaca oleh sensor, tegangan yang diatur secara *default* pada nilai 220 V, nilai daya listrik dan KWh penggunaan listrik yang telah dihitung sebelumnya oleh Websocket *server*, waktu data *monitoring* dimasukkan



ke *database server*, dan variabel *reset* yang digunakan untuk membedakan bulan saat ini dengan bulan sebelumnya.

### 3. Tabel ruangan

Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
SELECT * FROM `ruangan`
```

☐ Show all | Number of rows: 25 | Filter rows: Search

+ Options

	id_ruangan	nama_ruangan
<input type="checkbox"/> Edit Copy Delete	9	Kamar Baihaqi
<input type="checkbox"/> Edit Copy Delete	10	Kamar Faisal
<input type="checkbox"/> Edit Copy Delete	11	Kamar Jack

**Gambar 5.25** Tabel ruangan di *database server*

Tabel ruangan dibuat untuk menyimpan data ruangan dari tempat dimana perangkat *monitoring* listrik diletakkan. Pada tabel ruangan yang sudah dibuat, terdapat ID ruangan dan nama ruangan seperti yang ditunjukkan pada Gambar 5.25.

### 4. Tabel TDL

Showing rows 0 - 8 (9 total, Query took 0.0014 seconds.)

```
SELECT * FROM `tdl`
```

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

	id_tarif	golongan	tdl	keterangan	status
<input type="checkbox"/> Edit Copy Delete	1	450	415	Subsidi	0
<input type="checkbox"/> Edit Copy Delete	2	900	586	Subsidi	0
<input type="checkbox"/> Edit Copy Delete	3	900	1352	Non-Subsidi	1
<input type="checkbox"/> Edit Copy Delete	4	1300	1467.28	Non-Subsidi	0
<input type="checkbox"/> Edit Copy Delete	5	2200	1467.28	Non-Subsidi	0
<input type="checkbox"/> Edit Copy Delete	6	3500	1467.28	Non-Subsidi	0
<input type="checkbox"/> Edit Copy Delete	7	4400	1467.28	Non-Subsidi	0
<input type="checkbox"/> Edit Copy Delete	8	5500	1467.28	Non-Subsidi	0
<input type="checkbox"/> Edit Copy Delete	9	6600	1467.28	Non-Subsidi	0

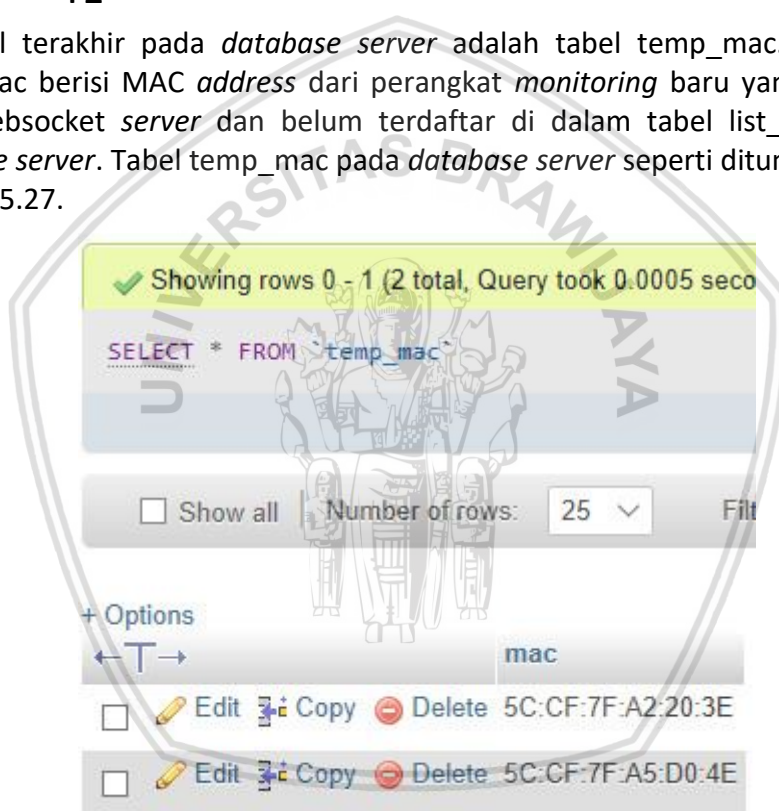
**Gambar 5.26** Tabel TDL pada *database server*



Tabel TDL menyimpan informasi terkait dengan tarif dasar listrik yang nantinya akan digunakan untuk menghitung estimasi biaya penggunaan listrik dari setiap perangkat *monitoring* listrik yang terpasang. Pada tabel tdl sesuai Gambar 5.26 terdapat ID tarif dasar listrik, golongan tdl yang merupakan kategori daya yang digunakan pada KWh meter (misal 450 VA, 900 VA, dan lain-lain), tdl yang merupakan tarif per KWh dari masing-masing golongan tdl, keterangan apakah golongan TDL termasuk dalam kategori golongan TDL bersubsidi atau non-subsidi, dan status yang merupakan golongan TDL yang dipilih pada *web monitoring* untuk menghitung estimasi biaya penggunaan listrik pada setiap perangkat *monitoring* listrik. Pada golongan TDL yang dipilih, maka akan memiliki nilai status 1, sedangkan lainnya memiliki nilai status 0.

## 5. Tabel temp\_mac

Tabel terakhir pada *database server* adalah tabel temp\_mac. Pada tabel temp\_mac berisi MAC address dari perangkat *monitoring* baru yang terdeteksi oleh Websocket server dan belum terdaftar di dalam tabel list\_device pada *database server*. Tabel temp\_mac pada *database server* seperti ditunjukkan pada Gambar 5.27.



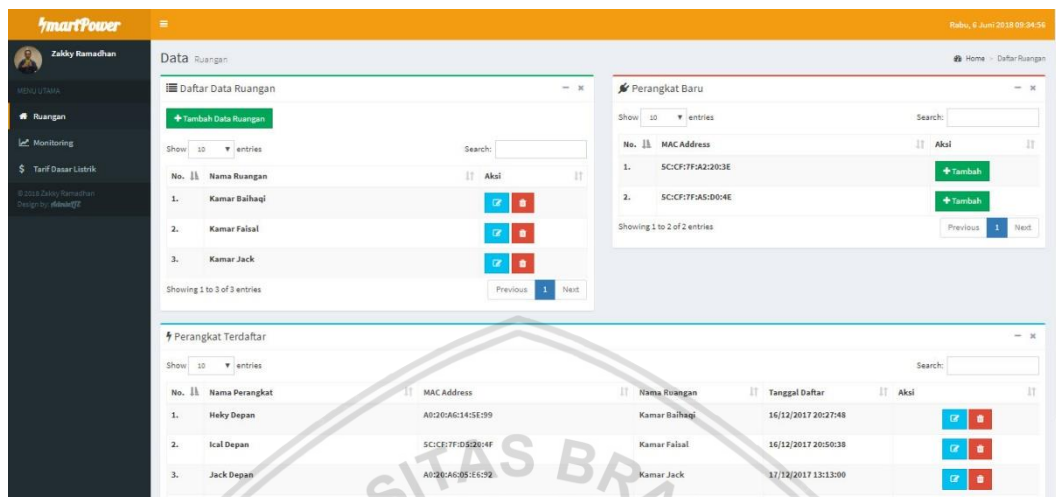
Gambar 5.27 Tabel temp\_mac pada *database server*

### 5.2.2.4 Implementasi Web Monitoring Listrik

Implementasi *web monitoring* listrik diterapkan pada *local server* dengan sistem operasi Microsoft Windows 10 64-bit sama seperti Websocket server dan *database server*. *Web server* yang digunakan pada penelitian ini adalah Apache dengan tampilan antarmuka *web monitoring* listrik menggunakan *template* dari AdminLTE berbasis *framework* PHP CodeIgniter. Untuk mengakses halaman utama dari *web monitoring* listrik dapat dilakukan dengan mengakses halaman **<http://localhost/skripsi>** di *browser* milik pengguna yang mana selanjutnya secara *default* akan diarahkan pada halaman **<http://localhost/skripsi/index.php/>**

**home/ruangan**. Terdapat beberapa halaman pada *web monitoring* listrik ini, diantaranya halaman ruangan, halaman *monitoring*, halaman detail *monitoring*, dan halaman tarif dasar listrik yang akan dijelaskan sebagai berikut:

## 1. Halaman ruangan

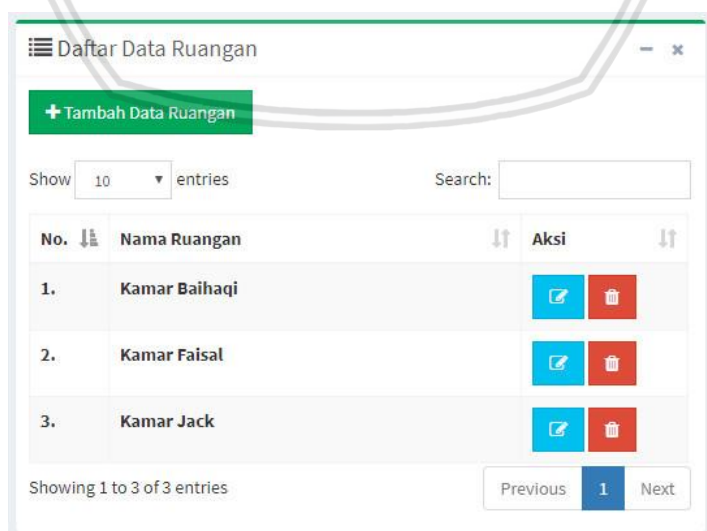


**Gambar 5.28** Halaman Ruangan

Halaman ruangan merupakan halaman *default* dari *web monitoring* listrik yang dibangun. Pada halaman ruangan terdapat tampilan daftar data ruangan, perangkat baru, dan perangkat terdaftar. Halaman ruangan dapat diakses pada alamat <http://localhost/skripsi/index.php/home/ruangan> seperti yang ditampilkan pada Gambar 5.28.

Pada halaman ruangan, pengguna dapat melakukan beberapa hal diantaranya:

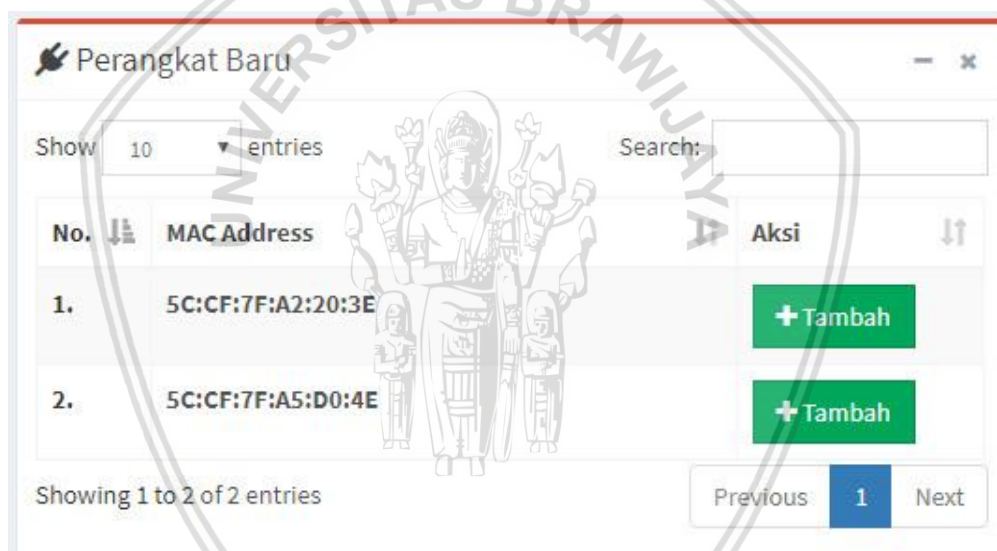
- Di kategori daftar data ruangan, pengguna dapat menambah ruangan baru dan mengubah serta menghapus ruangan yang sudah ada, seperti yang ditampilkan pada Gambar 5.29.



**Gambar 5.29** Daftar Data Ruangan

Ketika pengguna ingin menambah ruangan baru, maka pengguna harus memilih *icon* “Tambah Data Ruangan” berwarna hijau, lalu memasukkan nama ruangan yang diinginkan. Apabila pengguna ingin mengubah nama dari suatu ruangan, maka pengguna dapat menekan *icon* berwarna biru dari ruangan yang dimaksud, lalu selanjutnya memasukkan nama baru yang diinginkan. Untuk menghapus salah satu ruangan, pengguna perlu menekan *icon* tempat sampah berwarna merah dari ruangan yang dimaksud. Data yang ditampilkan pada kategori daftar data ruangan berasal dari tabel ruangan di *database server*.

- b. Setiap perangkat baru yang terdeteksi oleh *Websocket server*, maka akan tampil pada kategori perangkat baru di *web monitoring* daya listrik. Daftar perangkat baru yang ditampilkan tersebut bersumber dari tabel *temp\_mac* pada *database server*, seperti yang ditampilkan pada Gambar 5.30. Apabila pengguna ingin mendaftarkan perangkat baru tersebut, maka pengguna perlu menekan *icon* tambah berwarna hijau, lalu memasukkan nama perangkat baru dan pilih ruangan dimana perangkat tersebut ditempatkan.



**Gambar 5.30** Daftar Perangkat Baru

- c. Perangkat *monitoring* listrik yang sudah terdaftar pada tabel *list\_device* di *database server* maka akan ditampilkan pada kategori perangkat terdaftar. Pada kategori perangkat terdaftar, pengguna dapat mengubah konfigurasi dari perangkat yang sudah didaftarkan atau menghapus perangkat yang sudah terdaftar, seperti yang ditampilkan pada Gambar 5.31. Untuk mengubah konfigurasi dari sebuah perangkat terdaftar, maka pengguna perlu menekan *icon* berwarna biru dari perangkat yang dimaksud, lalu dapat mengubah konfigurasi nama perangkat dan ruangan dimana perangkat tersebut ditempatkan. Pengguna juga dapat menghapus perangkat yang sudah terdaftar dengan menekan *icon* tempat sampah berwarna merah dari perangkat yang dimaksud.

No.	Nama Perangkat	MAC Address	Nama Ruangan	Tanggal Daftar	Aksi
1.	Hecky Depan	A0:20:A6:14:5E:99	Kamar Baihaqi	16/12/2017 20:27:48	[Edit] [Delete]
2.	Ical Depan	5C:CF:7F:D5:20:4F	Kamar Faisal	16/12/2017 20:50:38	[Edit] [Delete]
3.	Jack Depan	A0:20:A6:05:E6:92	Kamar Jack	17/12/2017 13:13:00	[Edit] [Delete]
4.	Jack Belakang	5C:CF:7F:D5:1F:D8	Kamar Jack	19/12/2017 11:09:11	[Edit] [Delete]

Showing 1 to 4 of 4 entries

Previous 1 Next

Gambar 5.31 Daftar Perangkat Terdaftar

## 2. Halaman *monitoring*

Halaman *monitoring* merupakan halaman yang menampilkan rangkuman *monitoring* dari seluruh perangkat *monitoring* listrik yang sudah terdaftar pada *database server*. Rangkuman *monitoring* yang ditampilkan terdiri dari nama perangkat *monitoring*, nama ruangan, total penggunaan listrik, TDL aktif yang dipilih, dan estimasi biaya penggunaan listrik. Halaman *monitoring* dapat diakses pada alamat <http://localhost/skripsi/index.php/home/monitoring> seperti yang ditampilkan pada Gambar 5.32.

No.	Nama Perangkat	Nama Ruangan	Total Penggunaan Listrik (KWh)	TDL	Estimasi Biaya (Rp)
1.	Hecky Depan	Kamar Baihaqi	0.04024 KWh	Golongan : 900 VA Rp1.352,00/KWh	Rp54,54
2.	Ical Depan	Kamar Faisal	0.00000 KWh	Golongan : 900 VA Rp1.352,00/KWh	Rp0,00
3.	Jack Depan	Kamar Jack	0.06863 KWh	Golongan : 900 VA Rp1.352,00/KWh	Rp92,79
4.	Jack Belakang	Kamar Jack	0.04201 KWh	Golongan : 900 VA Rp1.352,00/KWh	Rp56,80

Showing 1 to 4 of 4 entries

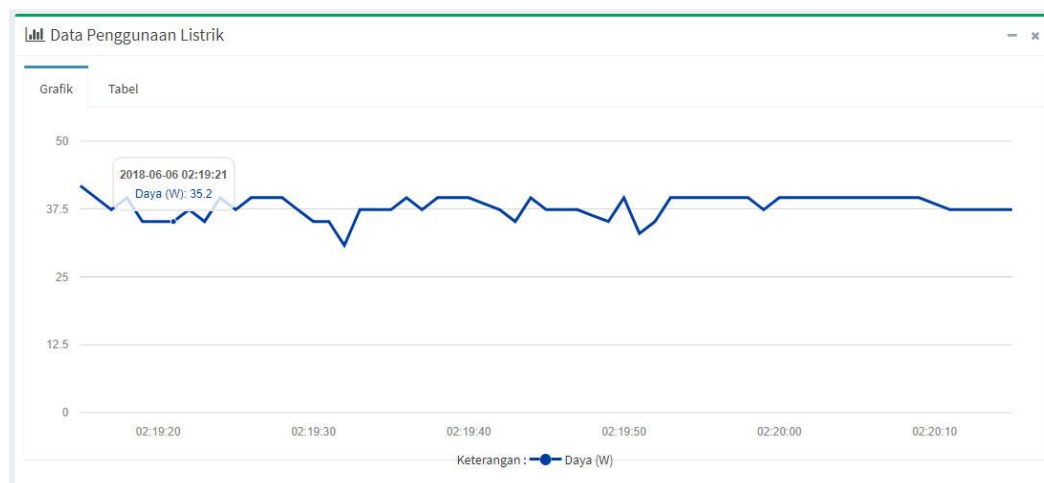
Previous 1 Next

Gambar 5.32 Halaman *Monitoring*

Apabila pengguna ingin melihat detail *monitoring* dari salah satu perangkat *monitoring* yang ada, maka pengguna dapat memilih perangkat *monitoring* yang ingin dilihat detailnya dan halaman akan berpindah ke halaman detail *monitoring* sesuai dengan perangkat *monitoring* yang dipilih.

## 3. Halaman detail *monitoring*

Halaman detail *monitoring* merupakan halaman yang menampilkan detail *monitoring* listrik dari satu perangkat *monitoring* listrik yang dipilih. Detail *monitoring* yang ditampilkan pada halaman detail *monitoring* berupa data grafik dan tabel dari data penggunaan listrik serta total penggunaan listrik dari perangkat *monitoring* tersebut berdasarkan rentang waktu yang dipilih. Secara default rentang waktu yang dipilih adalah data *monitoring* listrik selama satu menit terakhir. Halaman detail *monitoring* dapat diakses pada alamat [http://localhost/skripsi/index.php/home/<MAC\\_address>](http://localhost/skripsi/index.php/home/<MAC_address>).



**Gambar 5.33** Grafik Penggunaan Listrik

Pada Gambar 5.33 merupakan tampilan dari grafik *monitoring* penggunaan daya listrik selama satu menit terakhir pada salah satu perangkat *monitoring* listrik. Pengguna dapat melakukan *hover* pada grafik untuk melihat detail waktu dan besarnya daya listrik pada satu detik.

The table displays a list of electricity usage records. The columns are 'No.', 'Waktu', and 'Daya (W)'. The table shows 10 entries. A watermark of Universitas Brawijaya is visible in the background.

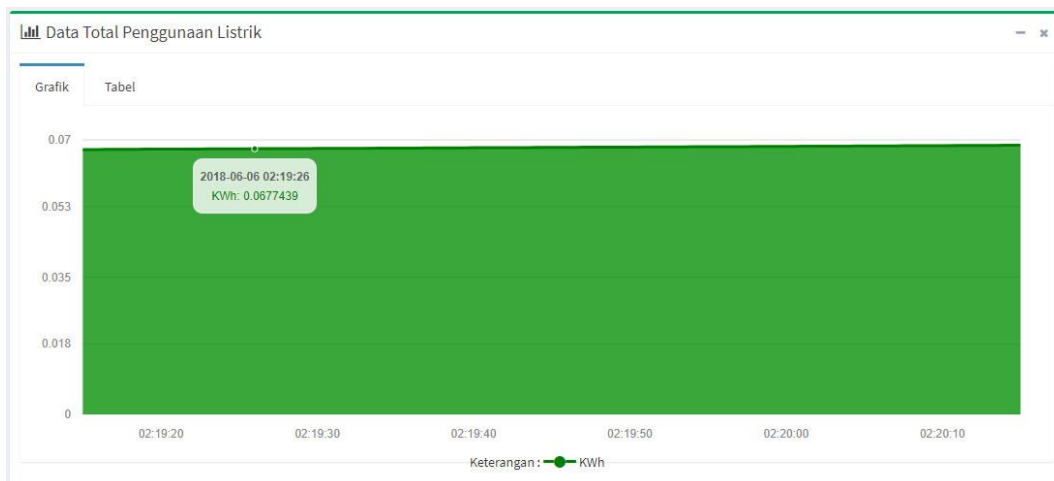
No.	Waktu	Daya (W)
1.	17/11/2017 13:13:21	173.8
2.	17/11/2017 13:13:22	83.6
3.	17/11/2017 13:13:23	72.6
4.	17/11/2017 13:13:24	68.2
5.	17/11/2017 13:13:25	72.6
6.	17/11/2017 13:13:27	70.4
7.	17/11/2017 13:13:28	70.4
8.	17/11/2017 13:13:29	63.8
9.	17/11/2017 13:13:30	63.8
10.	17/11/2017 13:13:31	66

Showing 1 to 10 of 7,003 entries

**Gambar 5.34** Tabel Penggunaan Listrik

Data penggunaan daya listrik juga ditampilkan dalam bentuk tabel seperti pada Gambar 5.34. Data yang ditampilkan terdiri dari waktu dan besarnya penggunaan daya listrik pada waktu tersebut.





**Gambar 5.35** Grafik Total Penggunaan Listrik

Untuk data total penggunaan listrik juga disajikan dalam bentuk grafik dan tabel seperti data penggunaan daya listrik. Seperti yang ditunjukkan pada Gambar 5.35, pengguna juga dapat melakukan *hover* pada grafik untuk melihat detail waktu dan total KWh penggunaan listrik pada waktu tersebut.

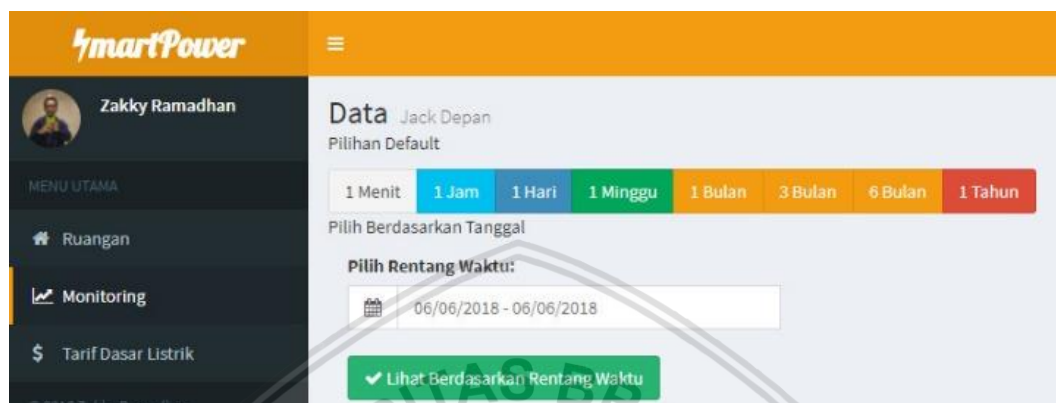
The figure is a table titled "Data Total Penggunaan Listrik". It displays 10 entries of electricity usage data. A large watermark of Universitas Brawijaya is visible in the background.

No.	Waktu	Total Penggunaan Listrik (KWh)
1.	17/11/2017 13:13:21	0.0603411
2.	17/11/2017 13:13:22	0.0603894
3.	17/11/2017 13:13:23	0.0604126
4.	17/11/2017 13:13:24	0.0604328
5.	17/11/2017 13:13:25	0.0604517
6.	17/11/2017 13:13:27	0.0604719
7.	17/11/2017 13:13:28	0.0604914
8.	17/11/2017 13:13:29	0.060511
9.	17/11/2017 13:13:30	0.0605287
10.	17/11/2017 13:13:31	0.0605464

**Gambar 5.36** Tabel Total Penggunaan Listrik

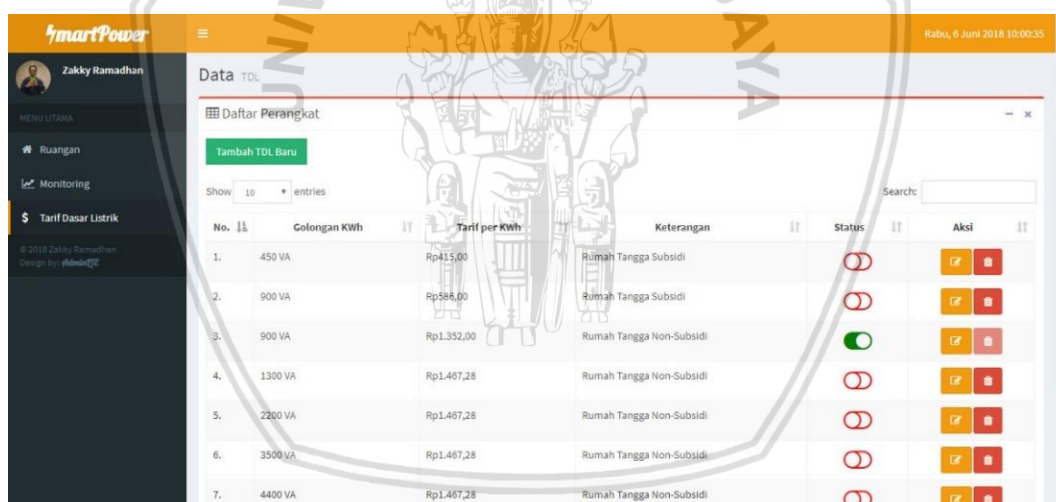
Pada Gambar 5.36 merupakan tampilan data total penggunaan listrik dalam bentuk tabel. Pada tabel tersebut terdapat waktu dan total KWh penggunaan listrik pada waktu tertentu.

Pengguna dapat mengatur rentang waktu *monitoring* daya listrik yang ditampilkan dengan memilih pilihan waktu yang tersedia pada halaman detail *monitoring*, seperti pada Gambar 5.37.



Gambar 5.37 Halaman Detail *Monitoring*

#### 4. Halaman tarif dasar listrik

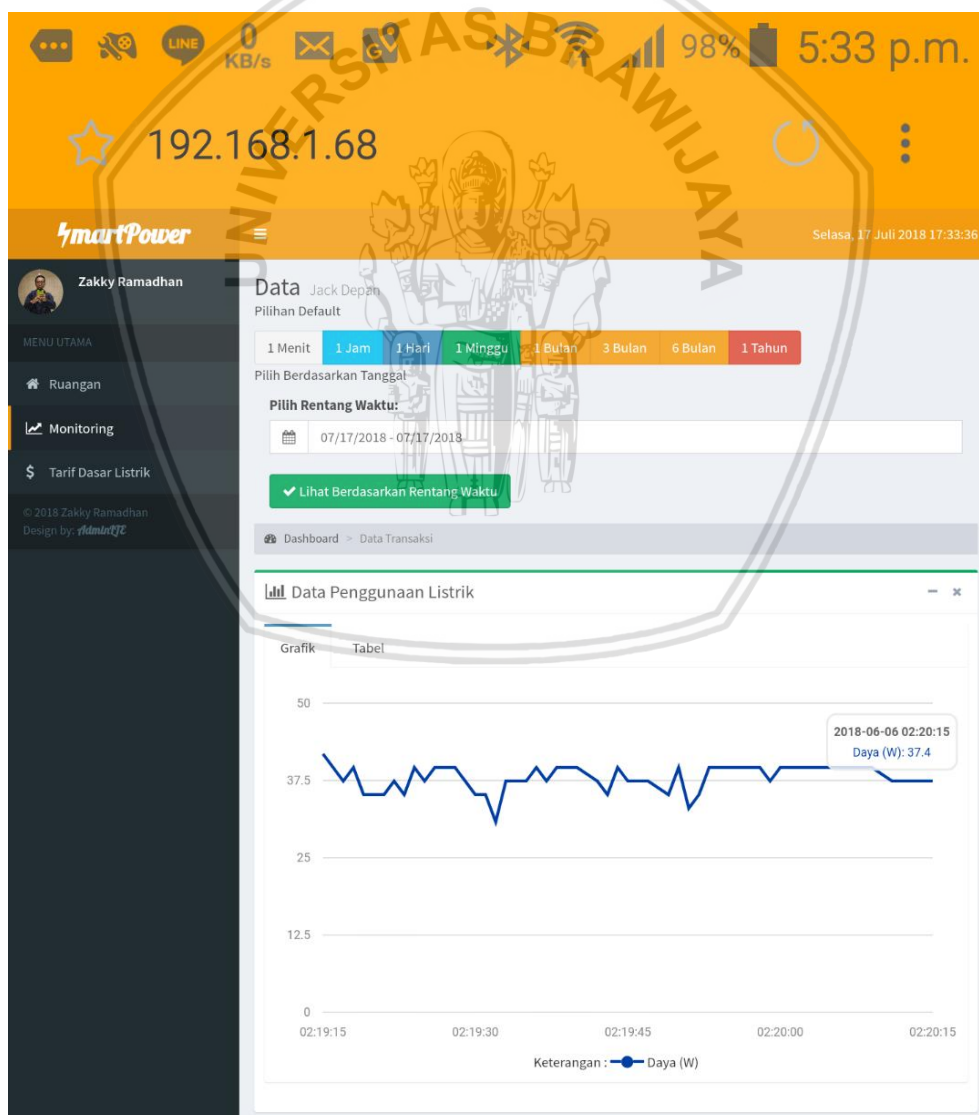


Gambar 5.38 Halaman Tarif Dasar Listrik

Halaman tarif dasar listrik merupakan halaman yang menampilkan daftar TDL. Daftar data TDL yang ditampilkan terdiri dari golongan TDL, tarif per KWh, keterangan apakah golongan TDL tersebut bersubsidi atau non subsidi, serta status TDL aktif yang dipilih untuk perhitungan estimasi biaya penggunaan listrik pada halaman *monitoring* dari tiap perangkat *monitoring* listrik. Untuk memilih TDL aktif, pengguna dapat menekan *icon slider* yang terdapat pada kolom status dari golongan TDL yang dimaksud. Pengguna juga dapat mengubah konfigurasi TDL dari suatu golongan TDL dengan menekan *icon* berwarna kuning atau menghapus suatu golongan TDL dengan menekan *icon* tempat sampah berwarna merah pada golongan TDL yang dimaksud. Tetapi, pengguna hanya dapat menghapus golongan TDL yang tidak memiliki status TDL aktif. Untuk menghapus

golongan TDL yang merupakan golongan TDL aktif, pengguna perlu menonaktifkan TDL aktif tersebut dengan menekan *icon slider* berwarna hijau, setelah itu pengguna dapat menghapus golongan TDL tersebut. Halaman tarif dasar listrik dapat diakses pada alamat <http://localhost/skripsi/index.php/home/tld> seperti yang ditampilkan pada Gambar 5.38.

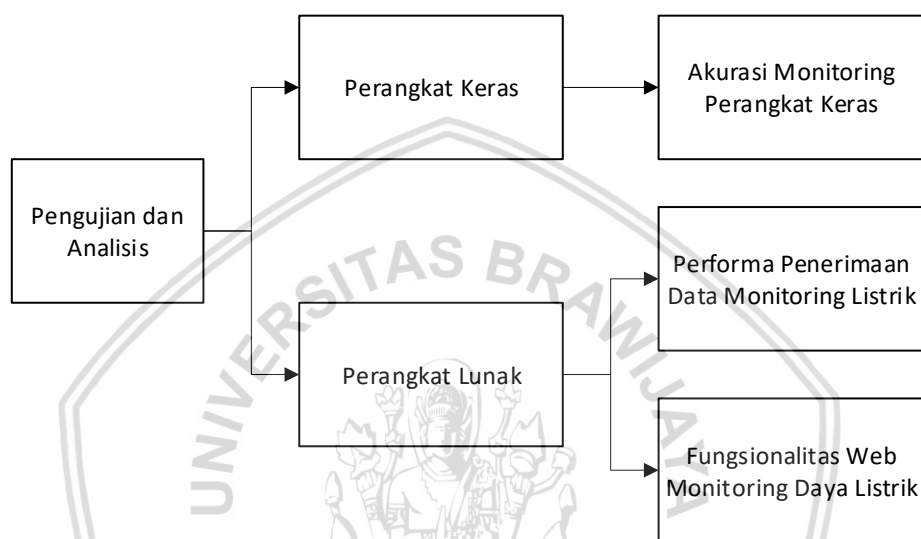
Pengguna juga dapat mengakses *web monitoring* daya listrik dengan menggunakan *browser* pada *smartphone*. Pada dasarnya proses untuk mengakses *web monitoring* daya listrik dengan menggunakan *browser* pada *smartphone* sama saja dengan mengakses menggunakan *browser* pada komputer *server*. Hanya saja alamat *localhost* diganti dengan alamat IP dari *web server* yang pada hal ini memiliki alamat IP 192.168.1.68. Untuk melakukannya, pengguna perlu mengakses alamat <http://192.168.1.68/skripsi/index.php>. Pada Gambar 5.39 menunjukkan *web monitoring* daya listrik diakses dengan menggunakan aplikasi Samsung *Internet Browser* pada *smartphone* Samsung *Galaxy Note 4* berbasis sistem operasi Android.



Gambar 5.39 Web Monitoring Daya Listrik Diakses Smartphone

## BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan dijelaskan mengenai pengujian dan analisa dari perancangan dan implementasi yang telah dilakukan sebelumnya. Pengujian dilakukan pada dua kategori pengujian, yaitu pengujian akurasi dari perangkat keras *monitoring* daya listrik dan pengujian perangkat lunak sistem *monitoring* daya listrik berupa pengujian performa penerimaan data *monitoring* dan pengujian fungsionalitas *web monitoring* daya listrik yang dibangun seperti pada *flowchart* pada Gambar 6.1.



Gambar 6.1 Flowchart Pengujian dan Analisa

### 6.1 Pengujian Perangkat Keras

Pada pengujian perangkat keras akan menguji akurasi dari pembacaan nilai arus listrik yang dilakukan oleh sensor arus listrik CT sensor YHDC SCT-013-000 dengan alat *power meter*. Selain itu, nilai tegangan 220 V yang diatur secara *default* pada program perangkat keras dan hasil perhitungan daya listrik juga akan dibandingkan dengan pengukuran yang dilakukan oleh *power meter*.

#### 6.1.1 Pengujian Akurasi *Monitoring* Perangkat Keras

##### 6.1.1.1 Tujuan

Pengujian akurasi *monitoring* perangkat keras dilakukan untuk menguji keakuratan hasil pengukuran arus listrik yang dilakukan oleh sensor arus YHDC SCT-013-000 yang terpasang pada terminal listrik di ruangan. Selain itu, nilai tegangan dan hasil perhitungan daya listrik yang didapat juga akan diukur akurasinya. Pengujian ini dilakukan dengan membandingkan hasil pengukuran arus pada perangkat keras dan tegangan serta daya listrik yang dihitung di Websocket server dengan menggunakan alat Taff *Energy Power Meter*.

Adapun ruangan yang diuji pada pengujian akurasi *monitoring* perangkat keras ini ditunjukkan pada Tabel 6.1.

Tabel 6.1 Ruang Pengujian

No.	Ruangan	Perangkat yang Terhubung dengan Terminal Listrik
1.	Kamar Baihaqi	Laptop Lenovo Ideapad G40-45, Kipas angin meja, <i>Portable Speaker</i> JBL, dan <i>Charger</i> ponsel iPhone 5V 1A.
2.	Kamar Faisal	Laptop Asus U36S, Kipas angin dinding, <i>Printer</i> Epson L100, <i>Speaker</i> aktif 2.0, LED <i>Monitor</i> LG 20MP38, dan <i>Charger</i> ponsel Sony 5V 2A.
3.	Kamar Jack	Laptop Dell Inspiron 14 7447, Kipas angin meja, <i>Printer</i> Canon E510, <i>Speaker</i> dbE NS77 2.0, LED <i>Monitor</i> Philips 227EQPH IPS, <i>Charger</i> ponsel Samsung 9V 1,67A, dan lampu tidur.

#### 6.1.1.2 Prosedur Pengujian

Prosedur pengujian yang harus dilakukan dalam menguji akurasi perangkat keras *monitoring* daya listrik ini diantaranya:

1. Pasang alat Taff *Energy Power Meter* pada stop kontak yang ada di ruangan.
2. Pasang terminal listrik ke alat Taff *Energy Power Meter*.
3. Pasang sensor arus yang ada pada perangkat keras *monitoring* daya listrik ke salah satu bagian kabel terminal listrik.
4. Hubungkan perangkat *monitoring* daya listrik tersebut dengan sumber listrik 5V dari *adapter* atau *power bank*.
5. Hubungkan *laptop server* dengan jaringan Wi-Fi.
6. Jalankan node.js *Websocket server*, Apache *web server*, dan MySQL *database server*.
7. Buka kode sumber program perangkat keras *monitoring* daya listrik di Arduino IDE, lalu *compile* dan *upload* program tersebut ke perangkat keras *monitoring* daya listrik.
8. Amati *output* arus listrik yang muncul pada *serial monitor* di Arduino IDE dan nilai perhitungan daya listrik yang ditampilkan pada *console log* node.js. Lalu bandingkan *output* tersebut dengan nilai pembacaan yang terlihat pada alat Taff *Energy Power Meter*
9. Catat hasil yang didapat dan buat analisa hasil pengujian.

#### 6.1.1.3 Hasil dan Analisis

Berikut merupakan hasil pengujian perangkat keras *monitoring* daya listrik yang digunakan. Akurasi didapat dengan membandingkan nilai hasil pengukuran yang didapat oleh perangkat keras dengan alat *power meter*.



Rumus Perhitungan Akurasi:

$$\text{Akurasi} = 100\% - \left( \frac{\text{nilai perangkat keras} - \text{nilai power meter}}{\text{nilai perangkat keras}} \times 100\% \right) \quad (6.1)$$

**Tabel 6.2** Pengujian Akurasi Perangkat Keras *Monitoring* Listrik

No.	Ruangan	Perangkat Keras			Power Meter			Akurasi		
		I	V	P	I	V	P	I	V	P
1.	Kamar Baihaqi	0,35 A	220 V	77 W	0,34 A	222 V	75,48 W	97,14 %	99%	98 %
2.	Kamar Faisal	0,42 A	220 V	92.4 W	0,4 A	222 V	88,8 W	95,24 %	99%	96,1 %
3.	Kamar Jack	0,6 A	220 V	132 W	0,57 A	222 V	126,54 W	95 %	99%	95,87 %

Rumus Perhitungan Daya Listrik:

$$P = V \times I \quad (6.2)$$

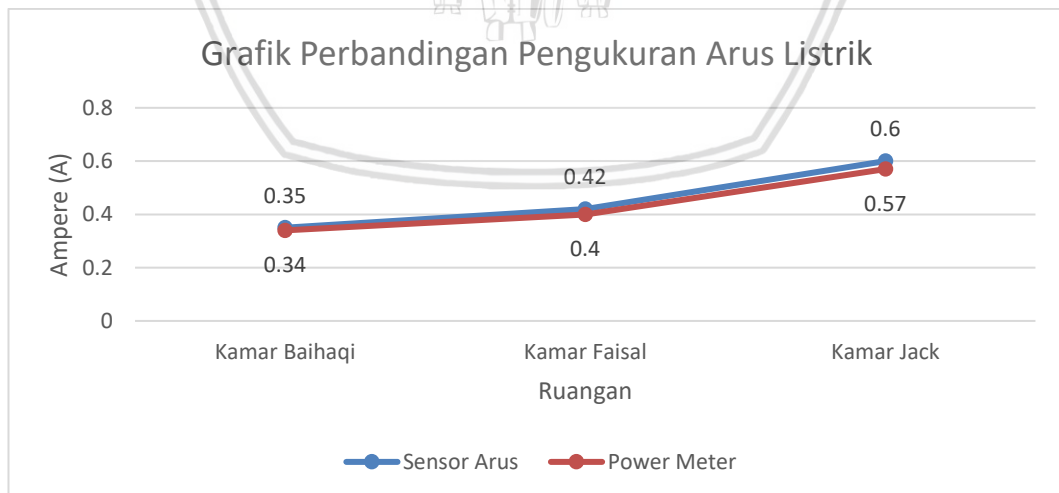
Keterangan:

I : Arus listrik

V : Tegangan listrik

P : Daya listrik

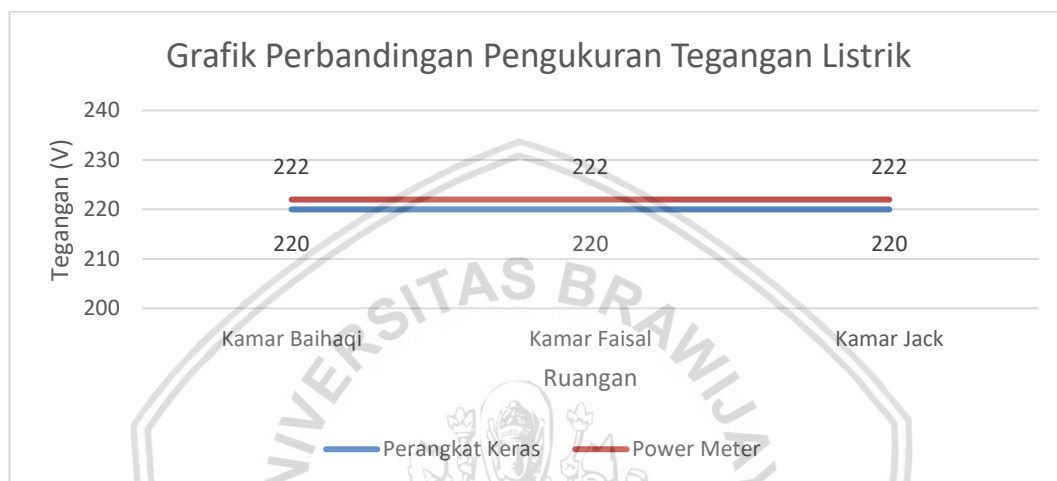
Tabel 6.2 memperlihatkan data hasil pengujian yang telah dilakukan. Penjelasan lebih lanjut mengenai hasil pengujian selanjutnya akan dijelaskan pada gambar grafik dibawah. Berikut pada Gambar 6.2 merupakan grafik perbandingan antara data arus listrik dari sensor arus dengan alat Taff *Energy Power Meter*.



**Gambar 6.2** Grafik Perbandingan Pengukuran Arus Listrik

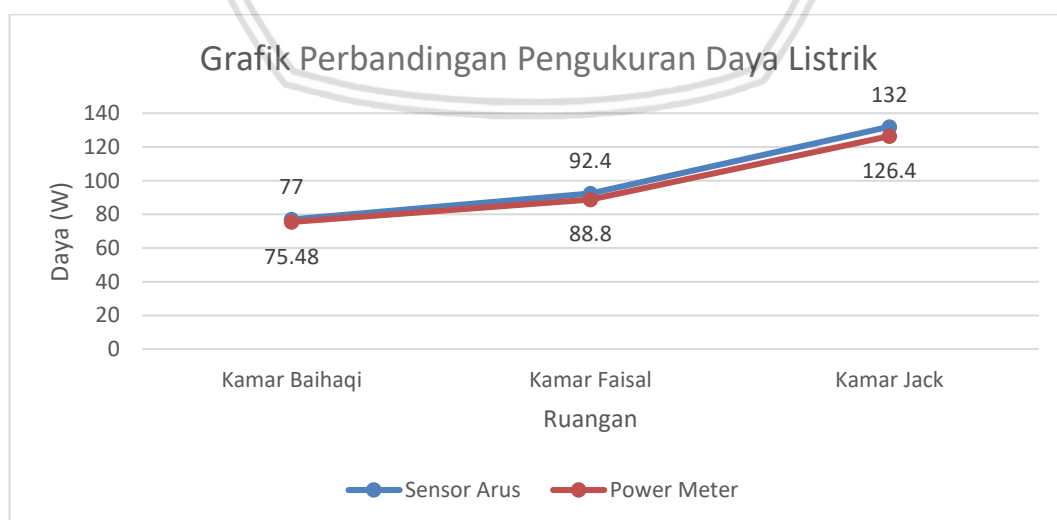
Pada grafik yang ditunjukkan oleh Gambar 6.2 dapat dilihat bahwa terdapat perbedaan data pengukuran arus listrik yang didapat antara sensor arus dengan *power meter*, namun perbedaan tersebut tidak terlalu besar. Perbedaan data

*monitoring* arus listrik ini disebabkan oleh penggunaan *burden* resistor pada perangkat keras yang tidak sesuai dengan perhitungan *burden* resistor. Perbedaan *burden* resistor yang digunakan tersebut dikarenakan nilai *burden* resistor sebesar  $23,3\ \Omega$  yang didapat pada hasil perhitungan merupakan nilai *burden* resistor yang tidak umum. Jadi, digunakan nilai *burden* resistor yang terdekat, yaitu sebesar  $22\ \Omega$ . Perbedaan tersebut juga dapat diakibatkan oleh *noise* yang terjadi pada proses *sampling* sinyal *analog* sensor arus listrik ke dalam bentuk nilai diskrit *digital* atau dari penggunaan komponen atau penyusunan rangkaian elektronik dari perangkat keras *monitoring* listrik yang kurang baik.



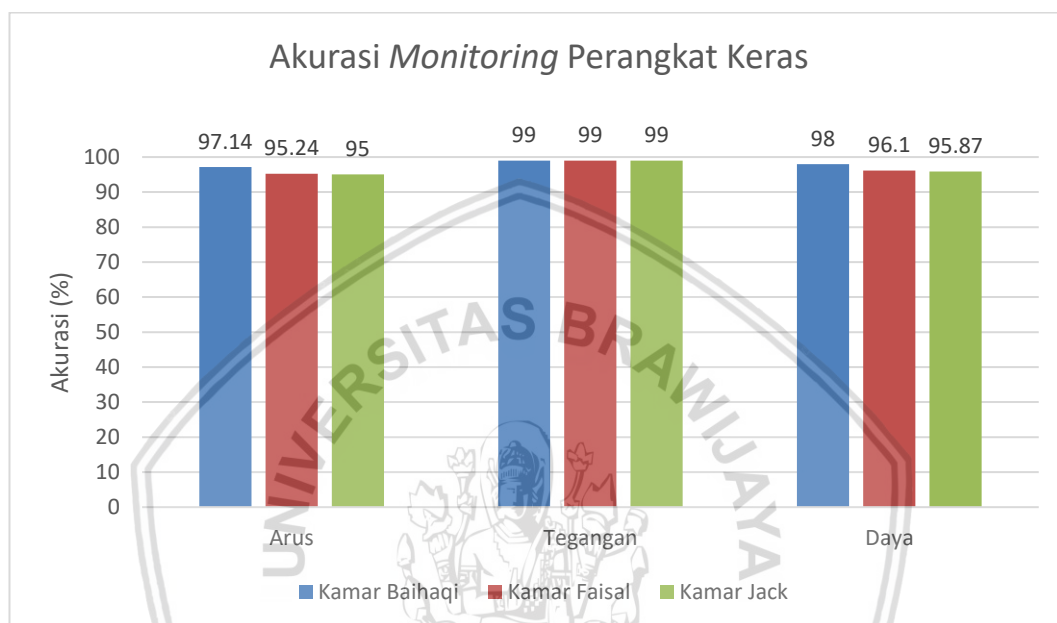
**Gambar 6.3** Grafik Perbandingan Pengukuran Tegangan Listrik

Sedangkan pada Gambar 6.3 untuk nilai tegangan listrik yang dibaca oleh *power meter* tidak berada tepat pada nilai tegangan sebesar 220 V. Jadi, terdapat perbedaan antara nilai 220 V yang diatur secara *default* pada program perangkat keras dengan hasil pembacaan alat *power meter*. Pengujian dilakukan sekitar pukul 15:00 WIB, dimana nilai tegangan yang didapat oleh *power meter* saat itu sebesar 222 V di tiga ruangan yang berbeda.



**Gambar 6.4** Grafik Perbandingan Pengukuran Daya Listrik

Pada Gambar 6.4 memperlihatkan grafik perbandingan perhitungan daya listrik yang dilakukan oleh perangkat keras dengan pengukuran alat *power meter*. Berdasarkan grafik pada Gambar 6.4, terdapat perbedaan antara perhitungan daya listrik yang dilakukan oleh perangkat keras dengan pengukuran *power meter*. Hal tersebut diakibatkan dari penggunaan nilai *default* 220 V pada nilai tegangan di program *monitoring* daya listrik, yang mana pada pengujian tegangan listrik nilai tegangan tidak berada tepat pada nilai 220 V. Jadi, hal tersebut akan mempengaruhi akurasi perhitungan daya listrik yang didapat.



**Gambar 6.5** Grafik Akurasi *Monitoring* Perangkat Keras

Berdasarkan grafik pada Gambar 6.5, untuk akurasi *monitoring* yang didapat oleh perangkat keras memiliki akurasi pengukuran yang berkisar antara 95 % hingga 97,14 % untuk arus listrik, 99% untuk tegangan listrik, dan 95,87 % hingga 98 % untuk akurasi perhitungan daya listrik.

## 6.2 Pengujian Perangkat Lunak

Pada pengujian perangkat lunak sistem *monitoring listrik* dilakukan untuk menguji fungsionalitas dari perangkat lunak yang dibangun apakah sudah berjalan sesuai dengan yang diharapkan. Pada pengujian ini akan dibagi menjadi dua kategori, yaitu pengujian performa penerimaan data *monitoring* listrik dan pengujian fungsionalitas *web monitoring* daya listrik.

### 6.2.1 Pengujian Performa Penerimaan Data *Monitoring* Listrik

#### 6.2.1.1 Tujuan

Pengujian performa penerimaan data *monitoring* listrik dilakukan untuk menguji waktu yang diperlukan untuk mendapatkan setiap data *monitoring*. Pengujian dilakukan mulai dari sensor arus listrik CT sensor YHDC SCT-013-000 melakukan akuisisi data sinyal listrik, data sinyal listrik diproses oleh

mikrokontroler NodeMCU dan dikirim ke Websocket server, hingga data *monitoring* listrik yang diterima oleh Websocket server disimpan pada *database server*.

#### 6.2.1.2 Prosedur Pengujian

Prosedur pengujian yang harus dilakukan dalam menguji performa penerimaan data *monitoring* listrik diantaranya:

1. Hubungkan *laptop server* dengan jaringan Wi-Fi.
2. Jalankan node.js Websocket server, Apache web server, dan MySQL database server.
3. Hubungkan perangkat *monitoring* daya listrik yang sudah dikonfigurasi dengan sumber listrik 5V dari *adapter* atau *power bank*.
4. Dengan menggunakan fungsi **this.getMilliseconds()** di node.js Websocket server amati detail waktu yang muncul pada *console log* node.js. Bandingkan selisih waktu dari tiap data yang diterima oleh Websocket server tersebut, lalu kurangi waktu tersebut dengan *interval delay* pengiriman data selama satu detik.
5. Buat analisa hasil pengujian.

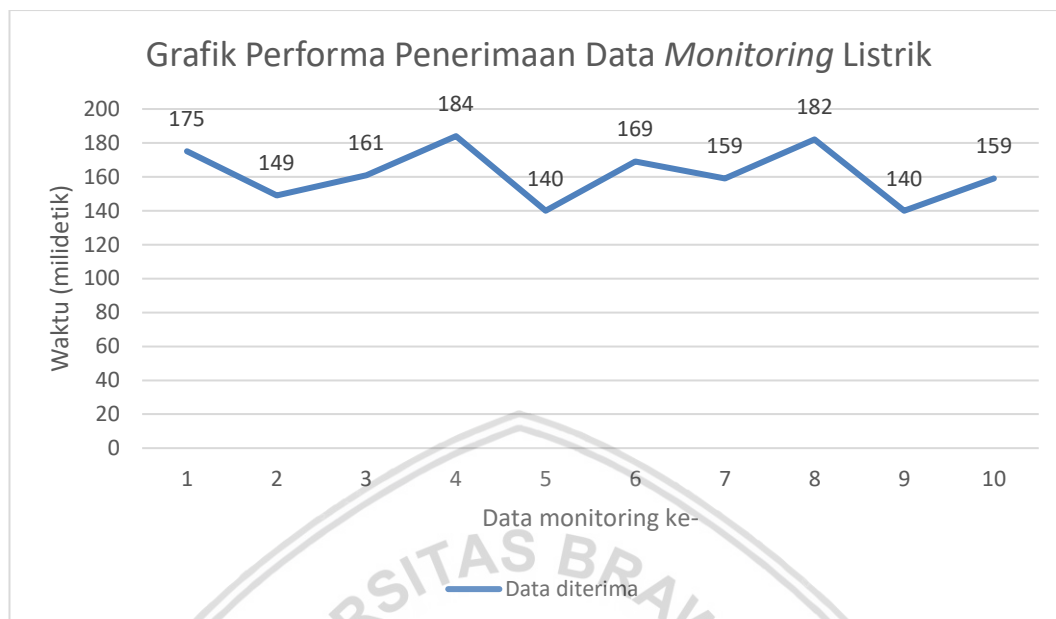
#### 6.2.1.3 Hasil dan Analisa

Berikut pada Tabel 6.3 merupakan hasil pengujian performa penerimaan data *monitoring* listrik pada sistem *monitoring* daya listrik yang dibangun.

**Tabel 6.3** Performa Penerimaan Data *Monitoring* Listrik

No.	Data <i>Monitoring</i> Diterima	Waktu yang diperlukan
1.	Data ke-1	175 milidetik
2.	Data ke-2	149 milidetik
3.	Data ke-3	161 milidetik
4.	Data ke-4	184 milidetik
5.	Data ke-5	140 milidetik
6.	Data ke-6	169 milidetik
7.	Data ke-7	159 milidetik
8.	Data ke-8	182 milidetik
9.	Data ke-9	140 milidetik
10.	Data ke-10	149 milidetik
Rata-rata		160,8 milidetik

Sedangkan grafik pada Gambar 6.6 dibawah merupakan representasi dari Tabel 6.3:



**Gambar 6.6** Grafik Performa Penerimaan Data *Monitoring* Listrik

Berdasarkan pengujian yang telah dilakukan, performa penerimaan data *monitoring* listrik dari sistem *monitoring* daya listrik yang dibangun memiliki performa yang baik sesuai harapan peneliti. Dimana dari 10 sampel data *monitoring* listrik yang diterima, sistem yang dibangun memerlukan waktu rata-rata sekitar 160,8 milidetik. Waktu tersebut merupakan waktu yang diperlukan oleh sistem dimulai dari sensor mengolah sinyal arus listrik hingga data *monitoring* listrik disimpan pada *database server*.

## 6.2.2 Pengujian Fungsionalitas *Web Monitoring* Daya Listrik

### 6.2.2.1 Tujuan

Pengujian fungsionalitas *web monitoring* daya listrik dilakukan untuk menguji seluruh fungsi atau fitur yang terdapat pada *web monitoring* daya listrik mulai dari tahap konfigurasi data ruangan, pendeteksian perangkat *monitoring* baru, konfigurasi pada perangkat *monitoring* yang terdaftar, menampilkan data *monitoring listrik* sesuai nama perangkat *monitoring* yang didaftarkan pada rentang waktu tertentu, dan pengubahan daftar nilai TDL. Seluruh fungsi atau fitur tersebut diuji apakah dapat berfungsi dengan baik sesuai dengan rancangan peneliti.

### 6.2.2.2 Prosedur Pengujian

Prosedur pengujian yang harus dilakukan dalam menguji fungsionalitas dari *web monitoring* yang dibangun diantaranya:

1. Hubungkan *laptop server* dengan jaringan Wi-Fi.



2. Jalankan node.js Websocket server, Apache web server, dan MySQL database server.
3. Hubungkan perangkat *monitoring* daya listrik yang sudah dikonfigurasi dengan sumber listrik 5V. Lalu, hubungkan juga perangkat *monitoring* daya listrik baru yang belum terdaftar di tabel perangkat terdaftar pada *database server*.
4. Akses halaman utama *web monitoring* listrik di *browser*.
5. Buat ruangan baru, ubah, dan hapus ruangan yang sudah ada di daftar ruangan.
6. Cek apakah ada perangkat baru terdeteksi pada bagian perangkat baru di *web monitoring* listrik. Jika ada, tambahkan perangkat baru tersebut ke perangkat terdaftar.
7. Cek bagian perangkat terdaftar, lalu ubah atau hapus konfigurasi salah satu perangkat terdaftar tersebut.
8. Buka halaman *monitoring* pada *web monitoring* listrik. Setelah itu, lihat rangkuman *monitoring* listrik yang ditampilkan.
9. Pilih salah satu perangkat *monitoring* daya listrik yang akan dilihat detail *monitoring* listriknya. Setelah itu, ubah rentang waktu *monitoring* yang ditampilkan.
10. Buka halaman tarif dasar listrik pada *web monitoring* listrik. Setelah itu, tambahkan TDL baru ke dalam daftar TDL, lalu pilih salah satu TDL yang akan diubah atau dihapus. Terakhir, ubah status TDL aktif ke golongan TDL lain.
11. Akses *web monitoring* daya listrik dari *browser* di *smartphone*.
12. Buat analisa hasil pengujian

#### 6.2.2.3 Hasil dan Analisa

**Tabel 6.4** Hasil Pengujian Fungsionalitas *Web Monitoring listrik*

No	Fungsi (Fitur)	Hasil
1.	Konfigurasi data ruangan	Berhasil
2.	Deteksi perangkat <i>monitoring</i> baru	Berhasil
3.	Konfigurasi pada perangkat <i>monitoring</i> yang terdaftar	Berhasil
4.	Tampilkan data <i>monitoring listrik</i> sesuai nama perangkat dan rentang waktu tertentu	Berhasil
5.	Pengubahan daftar nilai TDL	Berhasil

Dari hasil pengujian berdasarkan Tabel 6.4, peneliti dapat menyimpulkan bahwa pada *web monitoring* daya listrik yang dibangun mampu melakukan fungsi-fungsi yang sudah dirancang oleh peneliti. *Web monitoring* daya listrik diantaranya mampu melakukan perubahan konfigurasi data ruangan, mendeteksi perangkat *monitoring* listrik baru, melakukan perubahan konfigurasi pada perangkat *monitoring* listrik yang telah terdaftar, menampilkan data *monitoring* listrik berdasarkan perangkat *monitoring* listrik yang digunakan pada rentang waktu tertentu, serta mampu melakukan perubahan pada daftar TDL yang digunakan untuk perhitungan estimasi biaya penggunaan listrik dari setiap perangkat *monitoring* listrik.

Selain itu, *Web monitoring* daya listrik juga dapat diakses dengan menggunakan *browser* pada *smartphone*. Sehingga dapat disimpulkan bahwa *web monitoring* daya listrik mampu berjalan sesuai dengan apa yang sudah peneliti rancang dan implementasikan sebelumnya.



## BAB 7 PENUTUP

Pada bab ini membahas beberapa kesimpulan yang telah diperoleh dari penelitian, serta beberapa saran yang dapat digunakan untuk pengembangan penelitian berikutnya.

### 7.1 Kesimpulan

Berdasarkan analisa hasil pengujian yang telah dilakukan, maka dapat ditarik beberapa kesimpulan diantaranya:

1. Akurasi pengukuran arus dan daya listrik yang dilakukan oleh perangkat keras *monitoring* daya listrik menggunakan mikrokontroler NodeMCU dan sensor arus listrik YHDC-SCT-013-000 memiliki tingkat akurasi yang tinggi sesuai harapan peneliti. Hal ini berdasarkan dari pengujian yang dilakukan, didapat hasil akurasi pengukuran sensor arus listrik berada di kisaran 95% hingga 97,14% untuk arus listrik, 99% untuk tegangan, dan 95,87 % hingga 98 % untuk daya listrik. Sedikit perbedaan akurasi pengukuran tersebut dapat disebabkan dari penggunaan *burden* resistor pada perangkat rangkaian perangkat keras yang tidak sama atau mendekati dengan *burden* resistor dari yang seharusnya. *Noise* yang terjadi pada saat *sampling* data sinyal *analog* dari sensor arus listrik juga mempengaruhi keakuratan dari *output* nilai arus listrik yang dibaca. Nilai *default* tegangan 220 V yang digunakan pada perhitungan daya listrik di program *monitoring* daya listrik juga berpengaruh pada keakuratan nilai daya listrik yang diukur, karena berdasarkan pengujian dengan alat *power meter* nilai tegangan tidak selalu berada tepat di angka 220 V. Faktor lain yang mempengaruhi keakuratan pengukuran sensor juga dapat diakibatkan dari penyusunan komponen elektronika yang kurang rapi atau penggunaan komponen elektronika yang kurang bagus.
2. Sistem *monitoring* daya listrik yang dibangun memiliki performa penerimaan data *monitoring* daya listrik yang baik. Berdasarkan pengujian yang telah dilakukan, rata-rata waktu yang dibutuhkan mulai dari proses akuisisi data arus listrik oleh sensor arus hingga data *monitoring* daya listrik disimpan pada *database server* membutuhkan waktu rata-rata selama 160,8 milidetik.
3. *Web monitoring* daya listrik yang dibangun juga memiliki kinerja fungsionalitas fitur-fitur yang sesuai rancangan peneliti. *Web monitoring* daya listrik mampu melakukan konfigurasi pada data ruangan, mendeteksi perangkat *monitoring* listrik baru, mampu melakukan konfigurasi pada perangkat *monitoring* listrik yang telah terdaftar pada *database server*, menampilkan data *monitoring* listrik dari setiap perangkat *monitoring* daya listrik yang digunakan pada rentang waktu tertentu, serta mampu melakukan perubahan pada daftar TDL yang digunakan untuk perhitungan estimasi biaya penggunaan listrik dari setiap perangkat *monitoring* daya listrik. Selain itu, *web monitoring* daya listrik juga dapat diakses dengan menggunakan *browser* pada *smartphone*.

## 7.2 Saran

Adapun beberapa saran yang peneliti berikan untuk pengembangan sistem *monitoring listrik* selanjutnya antara lain:

1. Perlu penelitian lebih lanjut untuk mengurangi *noise* yang terjadi pada perangkat keras *monitoring* daya listrik. Karena terdapat perbedaan selisih pengukuran antara sensor arus dengan alat *power meter* sebesar 0.01 A hingga 0.03 A. Beberapa hal yang dapat dilakukan diantaranya dengan cara menggunakan *burden* resistor yang 100% sama dengan yang seharusnya sesuai dengan perhitungan *burden* resistor, mencetak PCB serta menyusun rangkaian elektronik dengan bantuan mesin, menggunakan komponen elektronika yang lebih berkualitas, atau penggunaan sensor arus lain yang lebih akurat.
2. Penggunaan mikrokontroler lain yang memiliki *input* analog lebih dari satu buah agar dapat menambahkan rangkaian untuk mengukur nilai tegangan listrik, nantinya diharapkan pengukuran daya listrik dapat lebih akurat.
3. Penambahan fitur pada *web monitoring* listrik seperti fitur *admin*, *login* pengguna, dan notifikasi *monitoring* listrik kepada pengguna.
4. Implementasi sistem *monitoring* daya listrik menggunakan protokol Websocket dengan Websocket *server* yang terhubung pada jaringan internet.
5. Diperlukan manajemen *database* yang lebih baik, karena rancangan *database* yang digunakan pada penelitian ini hanya berupa *database* sederhana. Jadi waktu *query database* yang dibutuhkan akan semakin lama seiring bertambahnya data *monitoring*.

## DAFTAR PUSTAKA

- Anggraeni, I., Ramdhani, M. & Ary Murti, M., 2016. *Sistem Monitoring Penggunaan Daya Listrik Menggunakan Sensor Arus Berbasis Mikrokontroler AVR ATmega 8535*, Bandung: Universitas Telkom.
- Chhetri, N., 2016. *A Comparative Analysis of Node.js (Server-Side JavaScript)*. Paper 5 ed. Minnesota: St. Cloud State University.
- Christina, B., 2012. *Hidupkan Kembali Gerakan Hemat Listrik 17-22*. [Online] Tersedia di: <<https://bisnis.tempo.co/read/news/2012/04/10/090396128/hidupkan-kembali-gerakan-hemat-listrik-17-22>> [Diakses 9 September 2017].
- Cloud Hosting Indonesia, PT., 2017. *Mengenal Apa itu Framework CodeIgniter*. [Online] Tersedia di: <<https://idcloudhost.com/panduan/mengenal-apa-itu-framework-codeigniter/>> [Diakses 1 Januari 2018].
- Darsiwan, 2016. *Apa itu WebSocket*. [Online] Tersedia di: <<https://www.codepolitan.com/menegtahui-apa-itu-websocket>> [Diakses 1 Januari 2018].
- Direktorat Jenderal Ketenagalistrikan, 2016. *Statistik Ketenagalistrikan*. 26 ed. Jakarta: Kementerian Energi dan Sumber Daya Mineral.
- Einstronic, 2017. *Introduction to NodeMCU ESP8266*, Kinabalu: Einstronic Enterprise.
- Gridling, G. & Weiss, B., 2007. *Introduction to Microcontrollers*. 1.4 ed. Vienna: Vienna University of Technology.
- Kho, D., 2016. *Rumus dan Rangkaian Pembagi Tegangan (Voltage Divider)*. [Online] Tersedia di: <<https://teknikelektronika.com/rumus-rangkaian-pembagi-tegangan-voltage-divider-resistor/>> [Diakses 1 Januari 2018].
- Kulkarni, A. S., Welch, PhD, K. C. & Harnett, PhD, C. K., 2011. A Review of Electricity Monitoring and Feedback Systems. *Proceedings of IEEE*, Issue 11, pp. 321-326.
- Kurniawan, A., 2017. *Pengembangan Sistem Monitoring Listrik pada Ruang Menggunakan NodeMCU dan MQTT*. Malang: Universitas Brawijaya.
- Le, B., Rondeau, T., Reed, J. & Bostian, C., 2005. Analog-to-Digital Converters. *IEEE Signal Processing Magazine*, 22(6), pp. 69-77.
- Listrik.org, 2017. *Tarif Dasar Listrik PLN Desember 2017*. [Online] Tersedia di: <<http://listrik.org/pln/tarif-dasar-listrik-pln/>> [Diakses 12 Januari 2017].
- Lutfi, F., 2017. *Mengenal Node.js*. [Online] Tersedia di: <<https://www.codepolitan.com/mengenal-nodejs-5880234fe9ae3>> [Diakses 4 Januari 2018].



- NodeMCU Team, 2014. *NodeMCU Connect Things Easy*. [Online] Tersedia di: <[http://nodemcu.com/index\\_en.html](http://nodemcu.com/index_en.html)> [Diakses 9 September 2017].
- OpenEnergyMonitor, 2017. *CT Sensor - An Introduction*. [Online] Tersedia di: <<https://learn.openenergymonitor.org/electricity-monitoring/>> [Diakses 9 September 2017].
- OpenEnergyMonitor, 2016. *About*. [Online] Tersedia di: <<https://openenergymonitor.org/?q=about>> [Diakses 2 Januari 2018].
- Sagar, 2016. *What is Correct Node.js Architecture*. [Online] Tersedia di: <<https://stackoverflow.com/questions/36766696/which-is-correct-node-js-architecture>> [Diakses 1 Januari 2018].
- Saha, M., Thakur, S., Singh, A. & Agarwal, Y., 2014. EnergyLens: Combining Smartphones with Electricity Meter for Accurate Activity Detection and User Annotation. *e-Energy '14 Proceedings of the 5th international conference on Future energy systems*, pp. 289-300.
- Sharma, A., Verma, A. & Bhalla, M., 2016. WiFi Home Energy Monitoring System. *2016 International Conference on Information Technology (IncITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds*, pp. 59-61.
- Shin, S., 2007. *Introduction to JSON (JavaScript Object Notation)*. 1 ed. Santa Clara: Sun Microsystems, Inc.
- Sim., A. X. A., 2014. *WebSocket*. [Online] Tersedia di: <<https://bertzzie.com/knowledge/javascript-lanjut/WebSocket.html>> [Diakses 1 Januari 2018].
- Sim, A. X. A., 2013. *Apa itu MVC?*. [Online] Tersedia di: <<https://bertzzie.com/knowledge/framework-php/1-Model-View-Controller.html>> [Diakses 1 Januari 2018].
- SparkFun Electronics, 2016. *Voltage Divider*. 1st ed. Niwot: SparkFun Electronics.
- Supriyono, H., Wahyudi, B. S. & Handaga, B., 2013. Saklar Lampu Otomatis dan Timer yang Dapat Diatur Untuk Menyalakan dan Memadamkan Sound System pada Persewaan Studio Musik. *Jurnal Emitter*, 13(2), pp. 1-8.
- Wicaksono, H., 2017. *Catatan Kuliah "Automasi 1"*, Surabaya: Universitas Kristen Petra.
- Wikipedia, 2017. *Power Strip*. [Online] Tersedia di: <[https://en.wikipedia.org/wiki/Power\\_strip](https://en.wikipedia.org/wiki/Power_strip)> [Diakses 9 September 2017].
- WWF Indonesia, 2016. *Earth Hour Indonesia*. [Online] Tersedia di: <[http://www.wwf.or.id/cara\\_anda\\_membantu/earth\\_hour\\_indonesia/](http://www.wwf.or.id/cara_anda_membantu/earth_hour_indonesia/)> [Diakses 9 September 2017].